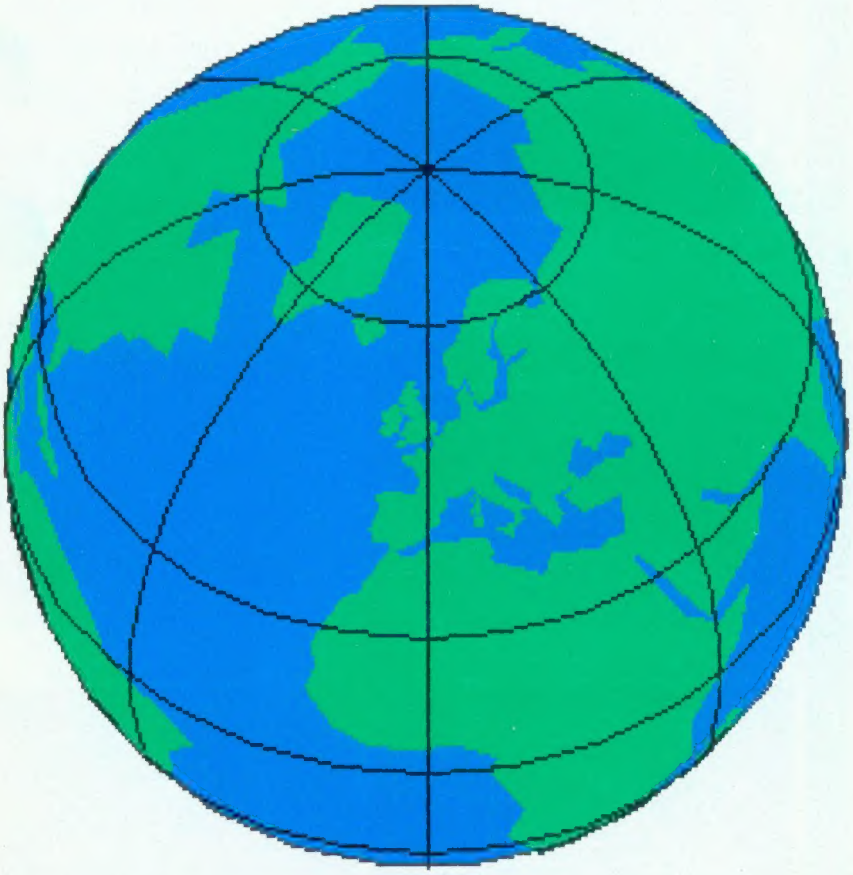
VOLUME 4 NUMBER 8 JANUARY/FEBRUARY 1986 PRICE £1.20

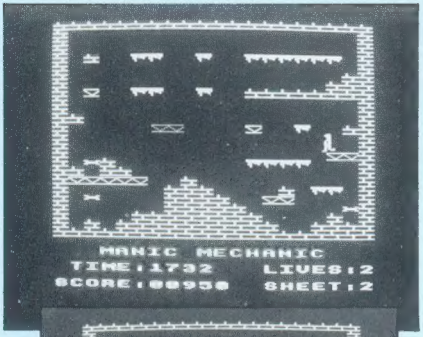
BEEBUG

FOR THE BBC MICRO

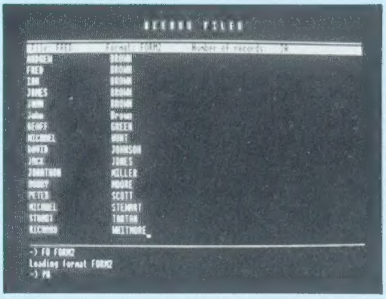


THE EARTH FROM SPACE

Manic Mechanic



Manic Mechanic



BEEBUG Filer

The Earth from Space

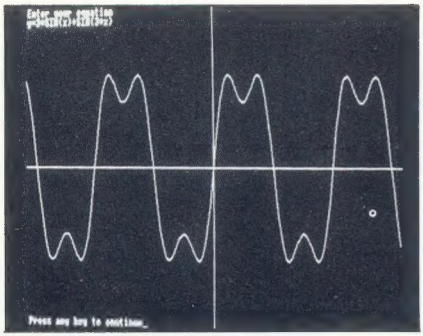
BEEBUG

VOLUME 4 NUMBER 8
JANUARY/
FEBRUARY 1986

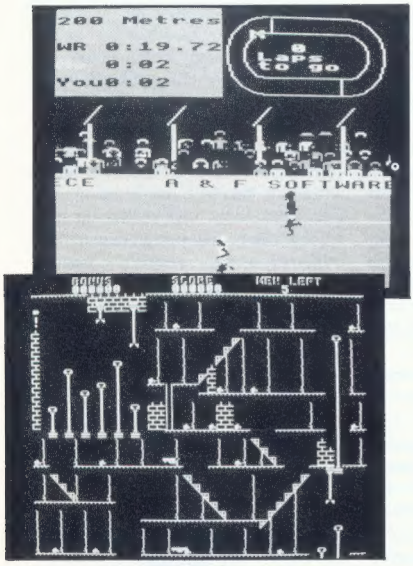
GENERAL CONTENTS

PROGRAMS

HINTS, TIPS & INFO



First Course



Games Reviews

Disc Recovery

```
Disc Rescue

'ESC' To Re-RUN          'R' To Recover file
'E' To Exit              'S' To Set start


. "THE KING","RO
CKY IV","JONATHA
N","DANGERMOUSE"
,"COMPO","BEAVER
","PENFOLD","JET
SET WALLY"....:.
....screen...4.
4,12,19,1,1;0;19
,5,5;0;19,8,6;0;
23;10,32;0;0;0;.
.C..1,28)"TIME:
";T%;.13,28)"LIV
ES:";Z%-1'"SCORE
:".(7)"SHEET:";P
%;....score(0):
.4,17,3.....tF
```

Track:1
Sector:6
Length:&1100
Abs. Sec:&10

Start Trk,Sec:,0,0

Disc Format:
80 Track
Single Density

EDITORIAL JOTTINGS

ACORN LAUNCH NEW BBC MICRO

The major news this month is the launch by Acorn of the long awaited successor to the BBC micro. In fact, it is not a single machine but a range of five systems, to be known as the Master series. There are no startling revelations, as the new machines build very heavily on the existing BBC micro, and Acorn have clearly decided that compatibility with existing machines is a high priority.

No doubt, many people will feel some disappointment that the Master series is not more innovative. However, we are all aware of the strengths of Acorn's model B and B+, and we should surely applaud the fact that much of what we value so highly has been retained. Looking at the specifications of the new range will quickly dispel any misgivings and show that Acorn have come up with a desirable and worthy successor to the highly acclaimed BBC micro.

Our report this month on the Master series is intended to be the first of several that we shall be featuring in the next few issues as we look in detail at what the new machines have to offer. At the same time, we shall continue to support existing Beeb owners just as strongly as ever. We hope that the Master series will be a major success for Acorn after their earlier financial troubles, and ensure Acorn's continuing support for all versions of the BBC micro for years to come.

BEEBUG MAGAZINE PROGRAM COLLECTIONS

Last month we also announced our own launch of two collections of programs from BEEBUG magazine. One collection consists of twelve of the most useful utility programs that have appeared in past issues of the magazine. The collection of six games features some which have previously appeared on the monthly magazine cassette/disc plus three more which are now published for the first time.

Both collections provide excellent value for money and are available on disc and cassette. Full details on both these collections are to be found in the supplement.

PROGRAM CLASSIFICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An unmarked symbol indicates full working, a single line through a symbol shows partial working (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that type of system. There is also a symbol for the B+ which includes the 128K version.

Basic I	I	Electron	
Basic II	II	Disc	
Tube		Cassette	
Model B+	+		

POSTBAG

POSTBAG

ROCKWELL SECONDED

I had intended to type in Dominique Willems' "Extended Assembler for the 65C02" (BEEBUG Vol.4 No.1), but before doing so I had a peek at the circuit board of my new second processor. Guess what? There lay a genuine Rockwell 65C02 and not the GTE 65C02 that the extension was written for! With yet another 32 instructions available, I'm wondering if the author could be persuaded to re-write the program to include these as well. I'm sure my second processor cannot be the only one with this device.

H.F.Lewis

We checked this with Acorn who say that Rockwell is sometimes used as a second source of supply for this chip, but that the number of second processors so fitted is and will remain only a very small proportion of the total. There is, therefore, no prospect of any further extensions being published in BEEBUG, but some readers (with a Rockwell chip) might like to try extending the original program themselves.

CAUGHT BY THE BINARY CHOP

I read with interest the article on searching and sorting in BEEBUG Vol.4 No.2 (June 1985), as recently I have needed to search for a record in a 2000 record file, and this method is certainly very fast. I would, however, like to point out a slight error in the fact that under certain circumstances (i.e. when the 'binary chop' ends up with a subscript of 3, the first record in the array is never found as a match, due to line 10090 st%>=fin%-2). If this line were to be changed to st%>fin%-2, then all should be well.

Mrs C.A.Pedder

WHEELS IN MOTION

Following your article in BEEBUG Vol.3 No.6, I have just started to make a graphics tablet. I have a hint that I think would be useful to other readers.

If you put a very small wheel under the end of the first arm (and perpendicular to its length) then this will support the arm and allow much greater freedom of movement. I found a wheel from an old Lego set to be ideal.

K.Chown

LOST IN THE WOODS

Your program on binary trees in BEEBUG Vol.4 No.5 has a flaw in it. It doesn't handle correctly the deletion of a right node which itself doesn't have a left branch. A simple example will show this to be so. Enter 'A,B,C' and then delete B. The right hand pointer of A is set to -1 (right%(0)=-1) and C is no longer 'seen' by the program.

This can be corrected by inserting the following line:

```
1415 IF turn$="R" AND
   left%(found%)=-1 THEN
        right%(lasturn%)=
        right%(found%):
        GOTO 1480
```

Apart from the above flaw, congratulations to the author of this program as it is very well written.

I.Wheatley

The two articles by Paul Ganney on Data Structures (BEEBUG Vol.4 Nos. 4 & 5) have prompted several favourable comments. However, Mr Wheatley is quite right in pointing out the program's failure to cater correctly for the special situation he describes.

CHRISTMAS RECURS LATE

The recursive trees program in the October BEEBUG (Vol.4 No.5) is a good demonstration of Beeb graphics; how about a snowflake design in a future issue? Enclosed is a screen dump of a Christmas tree with parameters; Twig 40, Tree 910, Angle 65, Limb 4, High 98 and Long 360.

Brian & Mary Hayes

This arrived just too late for our December issue but we still couldn't resist it. As for the snow, we are still waiting (maybe for the ingenuity of you, the readers).

BEEBUG SOFT FORUM

Masterfile II Auto-Startup options

Owners of Masterfile II may be interested to know that it is possible to customise the program so that it will automatically boot-up in the option of your choice, with various values pre-set as required by the user. If you examine the Masterfile program "LF", you will find the variables shown below together with their default values. These variables have functions as follows:

```
D% Tag file drive number
E% Tag directory
G% Left margin
L% Line length

$&C00 Today's date
$&C20 File name
$&C30 Menu option entered at startup
$&C40 Printer control code string
*DRIVE Masterfile II data drive
```

You may alter the values as required and re-save the file to disc under the same filename. (When amending the program we advise you to work from a back-up copy of the disc).

Technical Support

When communicating with our technical support department, it would help us greatly if you can supply full information about your machine, all Roms in it and the version number of the program relating to the enquiry. Rom information is displayed by typing *HELP , and disc information by typing *CAT.

RomIt Auto-Boot

A new and interesting feature has been included in RomIt which is not documented in the RomIt manual. Any Rom created by RomIt may be individually auto-booted by allocating it a unique letter, e.g. "A", and then pressing Shift/A/Break (in this example). This would cause the appropriate Rom to run automatically. Consequently, you can now have up to 16 Roms in your computer, each of which may be individually auto-booted as needed with Shift/A/Break, Shift/B/Break etc.

To set up this feature, RomIt users just need to save the "!BOOT" file, as described in the manual, but with the filename suffixed by the required letter. In our example this would be "!BOOTA". They should then issue the command *RBOOT which will put the necessary code into the Rom before it is blown.

Tailored Wordease (disc version) Auto-Boot

Wordease boots up to a Main Menu from which you may choose a variety of separate utility routines. As each is selected it is automatically loaded into segment 8 and run.

It is possible to create a special boot file to automatically load a selection of routines into predetermined segments, which will remove the necessity of selecting them from the menu each time.

To do this, first make a backup of your Wordease disc, as described in Appendix I of the Wordease manual, and then *BUILD your new !BOOT file in a similar fashion to the following:

```
*BUILD !BOOT
0001 *WORDWISE
0002 *KEY0 ...
0003 *KEY1 ...
0004 :SELECT SEGMENT 3
0005 :LOAD TEXT "W.H"
0006 :SELECT SEGMENT 4
0007 :LOAD TEXT "W.I"
0008 :SELECT SEGMENT 9
0009 :LOAD TEXT "W.UTILS"
0010  <Shift-f9> <Shift-f9> <Shift-f9>
0011 <Escape>
```

Note that lines 2 to 3 can be any function key definitions, lines 4-7 can be any segment selection followed by the loading of a Wordease routine, and lines 8-9 must select segment 9 in order to load the Main Menu. Line 10 requires you to press Shift-f9 three time to run the Main Menu, and pressing Escape in line 11 completes the *BUILD.

You should note that some Wordease routines have been specially tailored to work from segment 8 and these should therefore be modified if they are to be run from a different segment. For example, the Copy Seg. To Seg. routine will not let you copy to segment 8 (for obvious reasons) and the Status Display will give 'interesting' results, if used unmodified from a different segment.

News News News News News News Ne

Servos with a Smile

Surge Electronics has launched a servo control interface for the BBC micro. Up to eight servo motors can be controlled from the Beeb's User Port with the software supplied. Alternatively you can write your own control programs using Surge's driver routines. Each servo can be accurately positioned to within 0.7 of a degree over 180 degrees. The motors can control anything from a simple plotter to a full robot arm. The Surge Servo Interface costs £24.95 from Surge, 7 Falconers Field, Harpenden, AL5 3EU.

The Worm Turns Up

Level 9 has followed its huge success of 'Snowball' and 'Return To Eden' with the third adventure game in the trilogy. 'The Worm In Paradise' is a text only adventure on the Beeb but boasts many new features. The game has a 1000 word vocabulary and can understand instructions such as 'Examine all but the helmet, dummy, and leotard and go East'. Paradise is a future state where amusement arcades proliferate and everything is privatised (even the Police force). The Worm in Paradise costs £9.95 and Level 9 is on 0494-26871.

Replications

Superior Software has launched Repton 2 (see review of Repton in Vol.4 No.4). This, it claims, is 'more than a sequel - a new experience.' Different it certainly is. Repton 2 has more screens, more diamonds to collect, separate puzzle pieces to find, meteor showers to avoid, safes to unlock, a host of adversaries, and so on. Superior Software reckons that it is pretty difficult and are offering the statutory competition for the first to complete it. The only thing that hasn't changed is the price. Repton 2 is £9.95 on cassette and £11.95 on disc. Superior Software is on 0532-459453.

Floppywiser

Floppywise, the disc utilities ROM reviewed in Vol.4 No.1, has been extended to a 16K ROM. Software Services has now incorporated over 40 utilities in the ROM including a full disc sector and track ID editor and an extensive machine, sideways, and second processor RAM editor. All the old routines are still there and improved in some cases. Despite this, the new Floppywise costs £29.95 just like the old one and an upgrade service is available at a cost of £15 plus £1 postage on return of your old ROM and manual. Software Services is on 051-427 7894.

AV for Auntie

BBC Soft has released three educational packages for the Beeb that break new ground in educational program techniques. The packages comprise a work book, disc, and audio cassette. The program and audio cassette are used together to form an interactive audio-visual presentation for the pupil. The three packages cover 'Computers at Work' - an introduction to the use of computers for 9-12 year olds - 'Uniformly Accelerated Motion' and 'The Mole Concept' - two O level science tutorials. All three packs retail at a cost of £19.95 each. BBC Soft is on 01-580 5577.

The Prettiest List

Astral Software has taken the idea of BEEBUG's Pretty List program (see Vol.3 No.9) to extremes. The 'Super Intelligent List Augmenting System' (or SILAS) will produce an extremely pretty list with all the frills. Multi-statement lines are split, all FOR-NEXT and REPEAT-UNTIL loops are indented and line overflows maintain the indent, keywords are highlighted, and so on. The listing can be produced on the screen or on a printer, and the SILAS code resides in disc workspace so it does not affect the program being developed. SILAS is available only on disc for £9.95 from Astral on 05097-4815.

Knight Moves

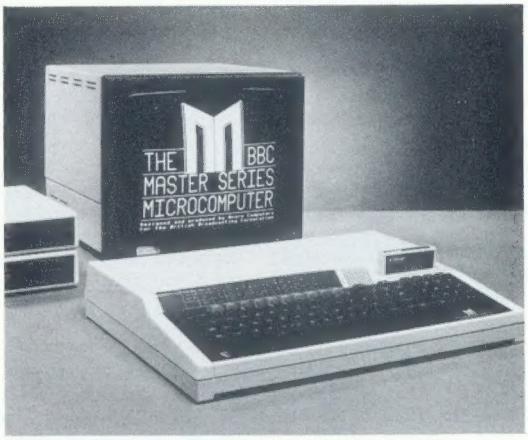
If BBC Soft's White Knight chess playing program (even mark 12) isn't advanced enough for you then there is now an upgrade available. The enhancement, 'Opening Knight' will enable you to print out all the moves played in a game of chess, and provide a wider opening move repertoire. Opening Knight costs £5.50 from Bernard Hill, Hawthorn Bank, Scott's Place, Selkirk, TD7 4DP.

ACORN LAUNCH NEW BBC MICRO

Acorn Computers have announced a new BBC Microcomputer System to replace the Model B. Known as the Master Series, this features a range of upgradable models. Mike Williams describes the new machines.

One of the best kept secrets of 1985 was revealed on 7th January when the successor to the BBC micro was revealed for the first time by Acorn at a major launch in London.

Acorn created a revolution in home computing when the BBC micro was first announced in 1981. Since then the BBC micro has been without doubt the most successful micro in its class in the UK and has achieved substantial success overseas, particularly in the educational market. At home, despite a popular image to the contrary, the largest proportion of BBC micros sold has been to the individual home user. Its further widespread popularity in education, business and industry has established the BBC micro has a first rate highly versatile machine with a wealth of software and hardware support.



Now, over four years on, Acorn have announced the successor to the original machine and its 64K AND 128K clones. The new BBC micro will be known as the Master Series and will comprise a range of five compatible and upgradeable systems. A major aim in developing the new series of machines has been to build on the strengths of the existing BBC micro. The basic design is an extension of the original, achieving a high level of compatibility, and externally even looking recognisably a BBC micro.

Maintaining compatibility will clearly please many existing owners, and Acorn will no doubt claim this as a strong point in favour of the new range. However, there will also be those who will say that Acorn have lost more than they have gained in so doing. Time will tell, and as is the case in other walks of life, the buying public may take a different view to that of the critics.

We expect to deal at greater length with the Master Series in future issues of BEEBUG, but we start with a basic run down of each of the new machines, plus some general comments on the new range and its compatibility with the old BBC micro.

THE MASTER SERIES

The new range of BBC micros can be summarised as follows (with full specifications at the end of this report):

MASTER 128

The basic model in the range with 128K of RAM and Edit, View and ViewSheet in one 128K ROM with extended operating system and Basic.

MASTER TURBO

Souped up version of the Master 128 with internal 6502 co-processor (Acorn's new name for second processor) and extra 64K bytes of RAM.

MASTER 512

The business oriented machine in the range with 16 bit co-processor, extra 512K bytes of memory, MSDOS compatibility and the GEM windows/icons/mouse environment.

MASTER SCIENTIFIC

The scientific version of the new machine with 32 bit co-processor, floating point processor, extra 512K bytes of memory, Panos Operating System and languages Fortran, Pascal, C, and 32 bit Basic.

MASTER TERMINAL

This is a cut down version of the Master 128 designed to function as an Econet terminal.

The new machines feature a much larger casing than the old, though still in the same 'Acorn' cream and with the same brown and orange QWERTY keyboard. The extra width accommodates a full 20 key numeric keypad, while the overall larger size means that co-processors and other upgrades can all be accommodated inside the main system box, instead of the multitude of external add-on boxes with the old Beeb.

Most of the original features have been retained, but as might be expected, much more software is now supplied in ROM format as standard; and all in one giant 128K ROM. The operating system includes the recently released Graphics Extensions as well as terminal emulation for when a machine is connected to a mainframe. Acornsoft's Basic Editor, View (word processor) and ViewSheet (spreadsheet) have all been included.

A disc interface is included as standard, based on the 1770 disc controller chip used in the model B+, and both the ADFS and 1770 DFS are included, again as part of the single 128K ROM.

There are several minor improvements, some of which will be very welcome. The Break key has been repositioned and made much more secure. The main circuit board includes a real time clock and this together with 50 bytes of battery backed RAM allows the machine to remember the current time and date, as well as other defaults. This information can be easily incorporated in any display or printout. An output socket for the sound system is also now a standard feature which will please many.

The use of sideways ROM software has become a major industry with the old model B and B+. With the Master Series, Acorn have made significant changes, which will not go down well with many existing users, and I suspect with many software houses supporting the Beeb market. 16K ROMs are out. The new machine has just three internal ROM sockets designed for 128K or 256K ROMs. Instead, the Master Series has introduced the dual ROM cartridge sockets featured on the Electron Plus 1. Well, it didn't succeed too well there and I am not convinced that it is the right move for this more sophisticated machine. It is the one feature that for me seriously lets down the visual appearance of the new machine. Jokes about ashtrays were rife when the BBC micro first appeared, and I fear Acorn may come in for some further stick on this.

The other models in the Master Series offer more power and memory. The Master Turbo clearly replaces the existing BBC micro with 6502 second processor. The Master 512 brings together some of the features seen on the ill-fated ABC series, including the windows/icons/mouse GEM environment for business users. The Master Scientific bears more than a passing resemblance to the Cambridge Scientific Workstation, and is clearly aimed at a similar market, although Acorn says that both new and old machines will continue.

The Master Terminal is a sensible approach to the network user, offering a version of the Master series but containing only the features needed to act as an Econet terminal.

IN CONCLUSION

There is a wealth of detail that cannot be covered in this first look at Acorn's brave successor to the much-loved BBC micro. This must wait for more detailed examination in future issues. On first impressions, the Master Series is a worthy if perhaps unexciting step forward. I am sure that many Beeb users, if they think about it, will acknowledge that the new machines provide much that they have been seeking for a considerable time. The strength of this approach is the relatively high level of compatibility between the old and the new. My criticsms are principally the new approach to sideways ROMs, and the 'trash can' on the keyboard. That said, Acorn have much to be proud of.

SPECIFICATIONS

THE MASTER 128

CPU 65C12 with 2MHz clock frequency

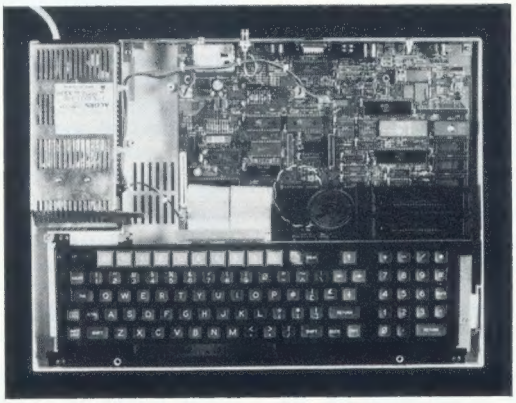
RAM 64K bytes of main memory
64K bytes of sideways RAM in four 16K byte pages including 50 bytes battery backed.

ROM 128K bytes
- 35K bytes Operating System with extended graphics and terminal software
- 16K bytes BBC Basic V4.0
- 16K bytes Edit (program and text editor)
- 13K bytes View V3.0 wordprocessor
- 16K bytes ViewSheet spreadsheet
- 16K bytes ADFS Advanced Disc Filing System
- 16K bytes 1770 DFS, B+ compatible

CARTRIDGE SOCKETS
- 2 enhanced Acorn cartridge sockets

INTERNAL ROM SOCKETS
- 2 x 128K or 256K bytes
- 1 x 128K bytes

DISC INTERFACE
- Supports double or single density, 40 or 80 track drives.
- Maximum capacity 1.28 Mbytes on twin double-sided 80 track drives.



OTHER INTERFACES
- As for model B/B+ except 1Mhz Bus uprated to 2Mhz Bus speed.
- External Tube Interface.
- Internal Tube Interface.

REAL TIME CLOCK
- Battery backed giving Time/Day/Date/Year.

KEYBOARD
- 63 key QWERTY keyboard
- 10 function keys, 20 key numeric keypad

SOUND
- 4 channels under software control
- Internal loudspeaker
- Phono output socket

THE MASTER TURBO

As for the Master 128 plus:

65C102 8 bit 4Mhz co-processor with 64K bytes of RAM and 4K bytes of ROM for Tube code, plus parallel processing support.

View automatically relocated on transfer from I/O processor memory. Hi-Basic and Hi-Edit on disc.

THE MASTER 512

As for the Master 128 plus:

80186 16 bit 8Mhz co-processor with 512K bytes of RAM and up to 256K bytes of ROM.

Acorn Mouse.

Digital Research DOS+ Operating System providing compatibility with MSDOS1.2 and CP/M86.

The GEM Collection from Digital Research with GEM Paint, GEM Desk Top and GEM Write.

16 bit BBC Basic.

Optionally Concurrent DOS 4.1 from Digital Research.

THE MASTER SCIENTIFIC

As for the Master 128 with the addition of: 32016 32 bit 8Mhz co-processor with 32081 floating point processor.

512K bytes of RAM, 16K bytes of ROM
Pandora operating system core
Tube communications code

Panos Operating System
Fortran 77
ISO Pascal Level 1
C programming language
Full 32 bit BBC Basic
32000 series macro assembler

THE MASTER TERMINAL

Processor and RAM as Master 128

ROM 64K bytes including:
- 32K bytes operating system
- 16K bytes BBC Basic
- 16K bytes Advanced Network Filing System

Network Interface
Cartridge Sockets

The Earth from space

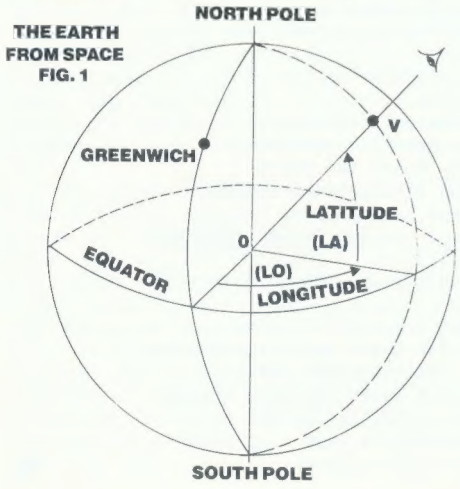
Michel Toulmonde describes how to re-create the world on your screen with this short Basic program.

This remarkably short program draws the Earth with all its continents as they are seen from any point in space.

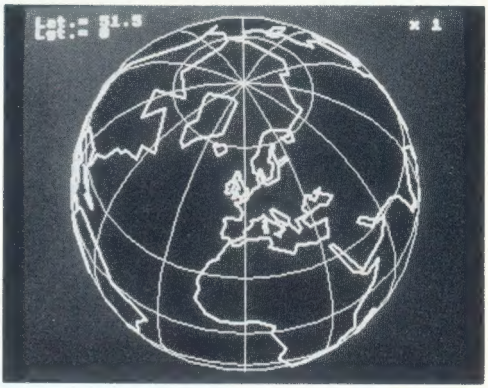
Take special care typing in the DATA statements and save the program before you try to run it. Disc users should set PAGE to &1200 (by typing PAGE=&1200<Return>), before loading or running this program. To choose your viewpoint above the Earth, you must provide the program with three numerical values - the co-ordinates of the observer in relation to the centre of the earth as shown in Fig.1. These are:

1. The distance of the observation point to the Earth's centre (O). This is conveyed in terms of the magnification of the drawing on the screen. The further into space, the smaller the earth seems. A value of 1 enables the whole globe to fit on the screen. Values above 10 or below 0.1 are accepted but the effect is poor.

2. The longitude (LO) of the point (V) on the Earth's surface directly below the observation point. This is the angle measured along the equator starting from the Greenwich Meridian, from +180 to -180 degrees (+ve west, -ve east).



THE EARTH FROM SPACE FIG. 1



3. The latitude (LA) of the point (V) on the Earth's surface. This is the angle measured on the line of longitude of point V, from -90 to +90 degrees (-ve south of the equator and +ve to the north).

A COMPLETE EXAMPLE

To observe the Earth with London directly below the observation point, select a magnification of 1 (to see the whole globe) by simply pressing Return. The co-ordinates of London are latitude 51.5 degrees north, longitude 0 degrees. Enter these as prompted.

In order to visualize the globe better, the program will also display lines of longitude (spaced as you choose) and the major parallels (the equator, the two tropics and the Arctic and Antarctic Circles). Select this option if you wish, when prompted. Now the Beeb does the rest; all you have to do is watch the screen.

It seems that nothing more is happening after the drawing of the American coasts. In fact, the continents, invisible in this case, situated on the other side of the Earth are being processed. Drawing all the countries takes about one minute and the program signifies that it has finished with a short beep. Pressing the space bar will now return you to the title screen to input the observation point for another view. To see how the other half lives enter the co-ordinates, -32,-117 to give the view of

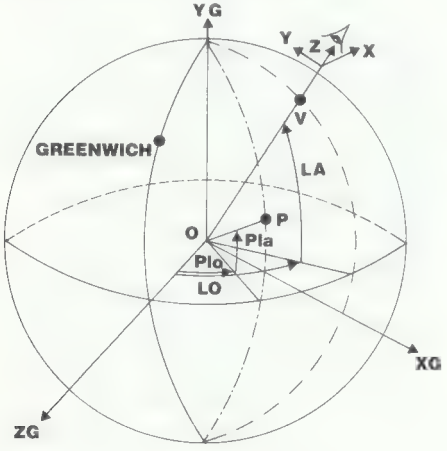
the Earth over Perth in Australia. Pressing Escape at any time will return you to the title screen.

You may wish to alter the value assigned to k in line 100 to produce a more circular globe depending on your own particular TV or monitor display.

THE DATA STATEMENTS

The co-ordinate data for the countries is contained in the DATA statements at the end of the program. These comprise sections for each continent or group of countries that can be drawn with an unbroken line. The first item in the data group is the number of data items in the group. This is followed by the latitude and longitude co-ordinate pairs that make up the outline of the continent. The last data item in the program is an end of data marker (-1000). This can usefully be placed earlier in the program as a check, while entering the data lines.

For example, you could run the program with an end of data statement at line 1755, having only entered the data for Great Britain, to check the operation of the program and the data for Great Britain before entering any more data.



THE EARTH FROM SPACE
FIG. 2

You can also, within the limits of the Beeb's memory, add further data to either increase the accuracy of the continental outlines already represented, or to add further islands of your own choosing.

THEORY

To obtain the drawing of the 3D globe on the 2D flat screen the spherical co-ordinate pairs must be projected onto a plane passing through the centre of the earth and perpendicular to the direction (OV) of the observation point.

Briefly, if Plo and Pla are the geographical co-ordinates (longitude and latitude) of a point P on the Earth's surface and R the radius of the earth (as shown in Fig.2), then the program will calculate the co-ordinates of P with regard to V (the point situated on our vertical and shown in the centre of the screen) by the following relationships:

```
X = R x COS(Pla) x SIN(Plo+LO)
Y = R x SIN(Pla) x COS(LA) - COS(Pla)
    x SIN(LA) x COS(Plo+LO)
Z = R x SIN(Pla) x COS(LA) + COS(Pla)
    x COS(LA) x COS(Plo+LO)
```

X and Y are the co-ordinates of the point, P, with respect to the centre of the screen and Z the distance 'behind' or 'in front of' the screen. If Z is negative then the point is situated behind the screen and not visible from the observation point and it is not drawn.

To complete the drawing of the Earth all that remains is to join the points calculated. The whole process of calculation and drawing is done by the procedure PROCc.

ADDING A PRINTER DUMP

A suitable machine code printer dump routine may be simply called at the right moment by adding this line to the program:

```
155 IF INKEY(-1) THEN <dump call>
```

The correct call (e.g. *GDUMP) for your dump routine should be substituted. Now pressing the Shift key along with the space bar at the end of a drawing will initialise your routine and dump the screen to your printer.

```
 10 REM PROGRAM GLOBE
 20 REM VERSION B1.0
 30 REM AUTHOR  M. TOULMONDE
 40 REM BEEBUG  JAN/FEB 1986
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 k=0.95:ON ERROR GOTO 180
110 MODE7:PROCinput
120 MODE1:PROCglobe
130 IF p<>0 PROClines
```

```
  140 PROCconts
  150 REPEAT UNTIL GET=32
  160 GOTO70
  170 :
  180 ON ERROR OFF
  190 IF ERR=17 THEN 70
  200 MODE 7:REPORT:PRINT" at line ";ERL
:END
  210 :
 1000 DEFPROCinput
 1010 FOR i=2 TO 3:PRINTTAB(12,i)CHR$130
CHR$141"E A R T H":NEXT
 1020 PRINT'"   This program draws a pic
ture of the"'"Earth as it would be seen
if we were"'"in space."
 1030 PRINT'" A magnification of 1 gives
 a complete"'"representation of the glob
e."
 1040 INPUT'" -Magnification : "R
 1050 R=500*ABS R:IF R=0 R=500
 1060 PRINT'" Give the positions of a po
int located"'"at the vertical of whichev
er country"'"or part of the earth you wi
sh to see:"
 1070 INPUT'" -Latitude  (-90 to 90):"
LA
 1080 IF ABS LA>90 THEN 1070
 1090 INPUT" -Longitude (180to-180):"L
O:LO=-LO
 1100 IF ABS LO>180 THEN 1090
 1110 PRINT'" -Meridian and parallel lin
es  (Y/N) ? ";
 1120 R$=GET$:PRINT R$
 1130 IF R$<>"Y" AND R$<>"y" THEN p=0:EN
DPROC
 1140 INPUT" -Angle step (eg 15,30,45,90
) : "p
 1150 IF p<1 THEN p=90
 1160 ENDPROC
 1170 :
 1180 DEFPROCglobe
 1190 VDU19,1,4;0;19,2,5;0;
 1200 VDU23,1,0;0;0;0;
 1210 VDU29,640;512;
 1220 IF R<550 COLOUR129 ELSE COLOUR 128
 1230 CLS:G=PI/180:S=SIN(LA*G):C=COS(LA*
G)
 1240 IF R>850 THEN T=0:GOTO 1340
 1250 GCOL0,0:XX%=R:YY%=0
 1260 MOVE XX%,YY%
 1270 FOR I%=0 TO 360 STEP 12
 1280 X%=R*COS(I%*G):Y%=k*R*SIN(I%*G)
 1290 MOVE 0,0
 1300 GCOL0,0:PLOT85,X%,Y%
 1310 GCOL0,2:MOVE XX%,YY%:DRAW X%,Y%
 1320 XX%=X%:YY%=Y%
 1330 NEXT:T=0
 1340 PRINTTAB(1,1)"Lat.= ";LA'" Lgt.= "
;-LO
 1350 PRINTTAB(35,1)"x ";R/500
 1360 ENDPROC
 1370 :
 1380 DEF PROClines
 1390 GCOL0,2
 1400 FOR L=0 TO 180-p STEP p
 1410 FOR F=90 TO 450 STEP12:PROCc(F,L):
NEXT
 1420 T=0:NEXT L
 1430 FOR L=0 TO 360 STEP 12:PROCc(66.5,
L):NEXT:T=0
 1440 FOR L=0 TO 360 STEP 12:PROCc(-66.5
,L):NEXT:T=0
 1450 FOR L=0 TO 360 STEP 12:PROCc(23.5,
L):NEXT:T=0
 1460 FOR L=0 TO 360 STEP 12:PROCc(-23.5
,L):NEXT:T=0
 1470 FOR L=0 TO 360 STEP 12:PROCc(0,L):
NEXT:T=0
 1480 ENDPROC
 1490 :
 1500 DEFPROCconts
 1510 RESTORE
 1520 GCOL0,3
 1530 REPEAT
 1540 READ N
 1550 IF N=-1000 THEN 1600
 1560 READ F,L:PROCc(F,L)
 1570 FOR i=2 TO N
 1580 READ F,L:T=1:PROCc(F,L):NEXT
 1590 T=0
 1600 UNTIL N=-1000
 1610 SOUND 1,-15,200,3
 1620 ENDPROC
 1630 :
 1640 DEF PROCc(F,L)
 1650 D=(L-LO)*G:Q=SIN D:M=COS D
 1660 U=COS(F*G):V=SIN(F*G)
 1670 X%=R*U*Q:Y%=R*(V*C-U*M*S)*k
 1680 IF T=0 THEN MOVEX%,Y%:GOTO 1710
 1690 Z=V*S+U*M*C
 1700 IF Z<0 THEN MOVE X%,Y% ELSE DRAWX%,Y%
 1710 T=1:ENDPROC
 1720 :
 1730 REM-------------------------------
 1740 DATA 25:REM=Great Britain
 1750 DATA 58.5,-5,58.5,-3,57.6,-4,57.7,
-2,56,-3,56,-2,53,.5,53,1.6,52,1.6,51.5,
.5,51.2,1.4,51,1,50,-6,51.4,-3.7,51.8,-5
.5,52.4,-4,53.3,-4.6,53.3,-3,54.8,-3.8,5
4.6,-5,55.5,-5,56,-6,57.5,-6,58,-5.3,58.
5,-5
 1760 DATA 12:REM=Eire-Northern Ireland
 1770 DATA 55,-6,55.2,-8.2,54.3,-8.3,54.
3,-10,53.4,-10,53,-9.2,52,-10.5,51.4,-9.
5,52.2,-6.2,54,-6.2,54.4,-5.4,55,-6
 1780 DATA 206:REM=Europe
 1790 DATA 71,27,70,19,64,10,62.5,5,58.5
,6,58,8,59,10.3,55.4,13,56,16,60,19,61,1
7,66,22,65.6,25,63,21,60,22,60.6,28,60,3
0,59,22,55,20,54,14
```

```
 1800 DATA 54.5,10,55.8,12.2,56,10,57.6,
10.3,57,8,54,8.3,53.3,5,51.4,3.6,50.9,1.
6,50.2,1.5,49.7,0.2,49.3,-0.1,49.4,-1.1,
49.8,-1.3,49.8,-2,48.7,-1.7,48.8,-3.1,48
.6,-4.7,48,-4.7,47.3,-2.5,46,-1.2
 1810 DATA 43.3,-1.5,43.7,-7.7,43,-9.3,4
1,-8.6,38.6,-9.6,38.6,-9,37,-9,37.1,-6.7
,36,-5.4,36.5,-4.8,36.8,-2,38.7,0.3,39.5
,-0.4,41.8,3.3,42.7,3,43.5,4,43.2,6.2,44
.3,8.9,43,10.5,41.3,13
 1820 DATA 40,15.7,38.9,16,36.6,15,38,12
.5,38,15.6,39,17,39.7,16.5,40.5,17,40,18
.5,42.5,14.1,43.6,13.6,44.4,12.3,45.5,12
.3,45.7,13.7,42,19.5,40.5,19.4,36.5,22.8
,38,24,40.8,23,41,29
 1830 DATA 43,27,47,31,46,33.5,45.5,32,4
4.3,34,46,37,46.5,35,47,39,46,37,42.5,42
.3,41,38,42,35,41,29
 1840 REM=Africa
 1850 DATA 40,26,37,28,37,36,31,34,31.6,
31,31,29,33,21,30,19,34,10,35,11,37,10,3
7,1,35,-2,35,-5,35.9,-5.4,35.8,-6,31,-10
,30,-10,28,-13,21,-17
 1860 DATA 17,-16,14,-17,8,-13,5,-8,5,-2
,6,4,3,10,-1,9,-11,14,-18,12,-35,19,-34,
26,-25,36,-20,35,-16,41,-5,39,4,48,12,51
,10,45,14,40,28,33
 1870 REM=Asia
 1880 DATA 28,35,12.5,44,18,56,23,60,24,
56,25,56,24,53,29.5,48,30,50,25,56,25,67
,21,72,12,75,8,77,10,80,6.5,80,6,80.5,6.
5,82,7,82
 1890 DATA 10,80,15,80,23,92,17,97,9,98,
4,101,1,104,5,103,9,99,13,100,8,105,11,1
09,15,109,19,106,22,108,21,110,23,117,30
,122,38,118,41,121
 1900 DATA 39,126,34,126,35,130,39,128,4
8,140,54,141,55,135,59,143,59,153,62,157
,62,163,57,156,51,157,55,162,60,163,60,1
70,63,180,66,177,67,190,70,175
 1910 DATA 72,130,74,110,77,112,72,70,69
,67,68,44,64,40,65,35,67,33,66.5,39,67.8
,41.5,71,27
 1920 DATA 7:REM=Iceland
 1930 DATA 66.5,-23,65,-24,66.5,-16,65,-
14,63,-19,64,-22,66.5,-23
 1940 DATA 10:REM=Corsica-Sardinia
 1950 DATA 43,9.4,42.4,8.5,41.5,9,41,9.5
,39,9.5,39,8.4,41,8.4,41.3,9.2,42,9.6,43
,9.4
 1960 DATA 6:REM=Madagascar
 1970 DATA -13,49,-17,44,-25,44,-25,47,-
15,50.5,-13,49
 1980 DATA 11:REM=Greenland
 1990 DATA 60,-44,65,-40,70,-22,82,-15,8
3,-30,78,-73,76,-68,70,-51,66,-54,61,-48
,60,-44
 2000 DATA 76:REM=N/S America
 2010 DATA 63,-77,52,-56,50,-65,46,-62,4
4,-70,42,-71,41,-74,35,-76,31,-81,27,-80
,25,-81,28,-83,30,-84,29,-90,27,-97,22,-
98,19,-97,19,-91,21,-90,21,-87
 2020 DATA 16,-89,15,-83,10,-83,9,-82.5,
10,-79,8,-77,11,-75,12,-71,11,-63,4,-52,
0,-50,-6,-34,-12,-39,-22,-41,-25,-48,-28
,-48,-41,-63,-51,-69,-55,-65,-55,-70
 2030 DATA -50,-76,-37,-74,-18,-70,-6,-8
1,0,-81,7,-77,9,-79,7,-81,10,-85,14,-90,
16,-95,15.5,-97,20,-106,22,-106,31,-113,
31.5,-115,30,-115,23,-110,25,-112,30,-11
6
 2040 DATA 34,-118,35,-121,39,-124,48,-1
25,59,-138,61,-148,54,-165,59,-158,62,-1
66,68,-167,71,-157,68,-110,70,-80,60,-95
,54,-80,63,-77
 2050 DATA 6:REM=Cuba
 2060 DATA 23,-82,22,-84,22.5,-82,20,-78
,20,-74,23,-82
 2070 DATA 5:REM=Haiti
 2080 DATA 20,-73,20,-70,18,-68,18,-74,2
0,-73
 2090 DATA 28:REM=Australia
 2100 DATA -10.5,142,-18,141,-15,136,-12
,137,-11,132,-15,129,-14,127,-20,120,-22
,114,-32,116,-35,115,-35,118,-32,130,-35
,135,-33,138,-35,138,-38,140,-39,143,-38
,145,-39,146
 2110 DATA -38,150,-34,151,-33,153,-29,1
54,-26,153,-20,148,-19,146,-10.5,142.4
 2120 DATA 23:REM=South Pole
 2130 DATA -63,-56,-66,-65,-73,-75,-73,-
100,-75,-100,-75,-137,-78,-160,-78,170,-
72,170,-66,135,-66,115,-67,90,-70,75,-68
,70,-66,55,-69,40,-71,20,-70,0,-71,-10,-
78,-35,-75,-60,-64,-59,-63,-56
 2140 DATA 21:REM=Japan
 2150 DATA 45.5,142,43.3,146,42,143,42.6
,141.6,40,140,38,139.5,37,137,35.5,136,3
5.5,133,33.5,129.5,31,130,31,131,33,132,
34,131,34.5,135,33.5,136,36,141,40,142,4
2,140,43.5,141.5,45.5,142
 2160 DATA 10:REM=Sumatra-Java
 2170 DATA 6,95,-4,102,-6,105,-8,115,-9,
115,-7,106,-3,106,0,104,5,98,6,95
 2180 DATA 6:REM=Borneo
 2190 DATA 2,109,7,117,5,119,-4,116,-3,1
10,2,109
 2200 DATA 10:REM=New Guinea
 2210 DATA 0,130,-2,138,-6,148,-11,151,-
8,144,-9,143,-8,138,-6,139,-4,133,0,130
 2220 DATA 13:REM=New Zealand
 2230 DATA -35,173,-37,176,-38,177,-37.5
,178.5,-41.6,175,-40.6,172.5,-43,171,-46
,166,-47,169,-40,175,-39,174,-38,175,-35
,173
 2240 DATA -1000
```

Out of the Shadows

Memory Expansion Boards Reviewed

Memory expansion is a popular solution to one of the more pressing problems for Beeb owners. Peter Rochford takes a look at the market leaders.

Product : B-32
Supplier : Aries Computers,
Science Park, Milton Road,
Cambridge CB4 4BH.
Tel: 0223-86214
Price : £92.00

Product : 32K RAM Card
Supplier : Watford Electronics,
250 Lower High Street,
Watford, Herts WD1 2AN.
Price : £70.15

The most often cited criticism of the BBC micro must be its shortage of user RAM when operating in one of the hi-res screen modes. There was no way around this problem until the introduction by Aries of the B-20 shadow RAM board (reviewed in BEEBUG Vol.2 No.9).

Shadow RAM systems like the B-20, and that featured in the new B+, provide an extra 20K or more of RAM in parallel with the computer's normal screen memory. By use of some clever software, the computer can switch between the two RAM areas in a way that is transparent to the user and with little loss in operating speed. The resultant system provides the user with 26K of RAM in any screen mode for Basic programs, or allows far larger data files in memory when used with such software as the View family and Wordwise.

Here we take a look at two new boards employing the shadow RAM principle. The B-32 from Aries is the latest design from them and follows on from the successful B-20. The other board is the 32K RAM card from Watford Electronics.

ARIES B-32

This is a larger board than the earlier B-20 and features 32K of RAM with a standard of construction that would be very difficult to fault.

Fitting requires the removal of the 6502 from the BBC, which is then placed in a socket on the B-32. The board is then plugged into the vacated 6502 socket on the computer. With the addition of the small support pillar provided, you have a very secure and stable installation.

Due to its size and the way it is fitted, the B-32 will not work with many ROM boards other than Aries' own B-12. If you decide to buy a B-32, it would be wise to check with Aries first whether it can be fitted with the ROM board you own. The B-32 cannot be used with the ATPL Sidewise board installed, though Aries will supply a kit to enable this configuration. Unfortunately this raises the keyboard and the lid of the Beeb by about 2 cm. There should be no problem, however, using the B-32 with the majority of double density disc controller boards.

Fitting is easy enough and the board can be up and running in less than five minutes. The ROM software that drives the board occupies a paged ROM socket on the B-32 itself and is supplied ready-installed.

At switch-on, the Beeb now greets you with the message 'BBC Computer 64K'. An improvement over the B-20 is that by means of an on-board link, the B-32 can be powered up with shadow RAM on or off. The B-32 has three main modes of operation:

20K shadow RAM + 12K sideways RAM
16K shadow RAM + 16K sideways RAM
two 16K banks of sideways RAM

Things don't end there however, as the controlling software for the B-32 lets you split the RAM up in many other ways too. For example, you could have 16K shadow RAM, 8K sideways RAM and an 8K printer buffer. There are many other combinations which the user can choose bearing in mind certain rules. Selecting and keeping track of the board's current configuration is achieved by a number of status screens. The buffer options of the B-32 allow any

of the Beeb's buffers to be extended to 4, 8, 12 or 16K depending on how else the board is configured.

The sideways RAM facility of the B-32 is a feature that many people will find attractive. Its appeal is heightened by the ROM/RAM management system in the B-32's software. This enables the loading of any RAM bank, either on the B-32 or B-12, and automatically write protects it. All ROMs in the machine can be disabled and re-activated at will and any sideways RAM bank wiped.

Further commands allow blocks of data to be moved between the B-32 RAM banks and the Beeb's RAM. This has a particular advantage of enabling long games programs, that won't work with the higher PAGE value of the ADFS, to be loaded and shifted down in sections from the banks of the B-32 after disabling the ADFS.

The manual and fitting instructions supplied with the B-32 although preliminary, were excellent. The final version of the manual will be more comprehensive with tutorials on all aspects of the board.

WATFORD 32K RAM CARD

This board is physically smaller than the Aries B-32 but also features 32K of RAM. The standard of construction and quality is again excellent.

Installation requires removal of the 6502 which must then be placed in a socket on the Watford board. Connection to the computer is effected by means of a ribbon cable and plug that attaches to the vacated 6502 socket.

An important advantage of this method of attachment, is that most ROM boards should work with the Watford board. I have used it successfully with the the Aries B-12 and the ATPL.

The software to drive the board is supplied on a ROM and plugs into a vacant paged ROM socket.

Operation of the board is straightforward and allows it to be configured in two ways:

20K shadow RAM + 12K extended buffer
32K extended buffer

The buffer may be any of the Beeb's buffers: printer, RS423 etc.

When the Beeb is powered up the shadow RAM is active, but a *RAMOFF command or Ctrl-Break will disable it. The software for the Watford RAM card provides a host of commands for selecting the various options and also a RAM editor that allows examination and modification of both the shadow RAM bank and the Beeb's RAM. The current configuration of the RAM card can be checked easily by a status screen.

A feature I particularly like with this board is the single command for entering Wordwise or View. This performs several operations with but one command to both configure the board and enter the wordprocessor.

Documentation supplied with the Watford board consists of a spiral bound manual that, although on the thin side, is well written.

CONCLUSION

Both of the boards performed faultlessly throughout the course of this review, and were a joy to use. They are excellent products that extend the power of the BBC and demonstrate that support for the machine is still strong.

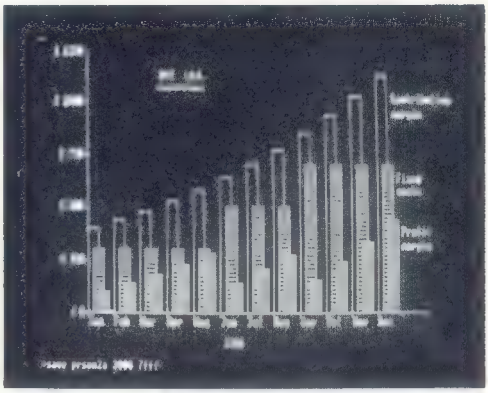
It is easy to dismiss the Watford board in favour of the Aries because of the number of extra operating options the B-32 provides. However, you should bear in mind the Watford board is £22 cheaper than the Aries and that it will allow most ROM boards to co-exist with it.

For many people the Watford board will provide an excellent solution to their need for extra RAM at a realistic price, whilst enabling them to retain their existing sideways RAM/ROM hardware.

However, there is no getting away from the fact that the B-32 is a remarkable product. The number of operating options of this board make it very appealing. The B-32 software sets new standards of user-friendliness, with its ROM/RAM management system, and controls the whole set-up beautifully. Coupled with an Aries B-12 ROM board containing 16K CMOS RAM (if your budget will stretch to it) you have an expansion system that is hard to better.

Inter-Sheet and Inter-Chart

Computer Concepts' Inter-sheet is not only a major new spreadsheet for the Beeb but with Inter-chart represents the start of a whole family of interlinked ROM software. David Otley has been checking them out.



Product : Inter-sheet and Inter-chart
Price : £56.35 and £36.80
Producer : Computer Concepts

Inter-sheet and Inter-chart are both available from BEEBUG at special offer prices of £52.35 and £29.90 – see the BEEBUGSOFT information in the supplement for full details.

INTER-SHEET

Inter-sheet is a high quality spreadsheet program, based on two ROMs totalling 24K, and is competitive with Viewsheet and Ultracalc 2 (see previous reviews in Vol.3 No.3 and Vol.4 No.1). It is capable of performing virtually all the calculations that can be undertaken by these other packages, although some functions are operated in a rather different manner. I shall therefore concentrate on the differences rather than the similarities.

Firstly, Inter-sheet has an introductory menu of a type that will be familiar to users of Wordwise. This allows the standard loading, saving and printing operations to be carried out from the screen menu, and also permits default options to be set for the screen display. A nice feature is that the user is warned when attempting to load a new sheet from disc, if another sheet is already in memory, and also when attempting to save a file with the same name as one already on disc. This alone may sell the product to those prone to forget such details!

The function keys are used as in Viewsheet to edit and format cell contents, to set the re-calculation mode, and to insert and delete rows and columns. However, just one function is allocated to each, allowing the user to program them, using Shift and Control, for his own requirements. In addition, one key is used to toggle the display between 40, 80 and 105 character modes, if available free memory permits.

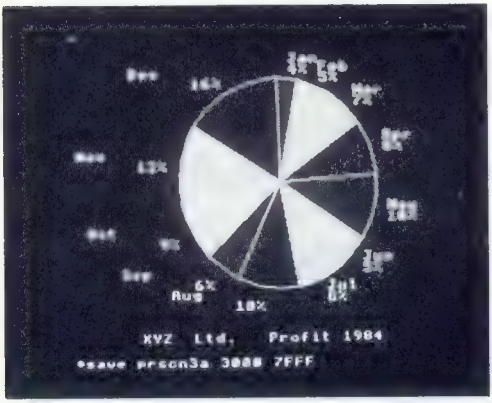
Yes, 105 characters can be displayed across the screen, using a specially defined character set in mode 0. The legibility of this depends on the quality of your monitor. I found it just readable on a standard quality Microvitec, and quite readable on all monochrome monitors. The 105 character display is of great value as it means that a 12 month budget complete with row labels and an annual total can be viewed on a single screen. The disadvantage is that such a display uses a great deal of memory, although this can be overcome by using a B-Plus or a screen memory expansion board.

All the usual commands are available, implemented in a form similar to Ultracalc (i.e. / followed by a single letter), giving an effect that can be applied to a single cell, a row, a column or the whole sheet. At each stage, user-friendly prompts are issued to guide the inexperienced user easily through the process. At first sight, the replication command (/COPY) seems inadequate as it allows only relative replication. Admittedly this is the most commonly needed method, but absolute replication can be obtained by using the BOX command in a formula to preserve absolute co-ordinates. However, I soon got used to this way of doing things and have now come to prefer it. A full range of functions, such as SUM, AVERAGE, MAX and MIN are available together with

all the Basic maths functions. In addition, conditional (IF) statements and look-up tables are provided.

The program is limited to row by row calculation, and can be put into manual re-calculation mode to save time whilst entering formulae on a large sheet (up to 255 rows and 64 columns is possible - although not both in 32K). Labels have to be entered preceded by quotation marks, and they can be centred as well as left and right justified. Column widths can be set individually and cells can be formatted in a choice of three ways including a fixed number of decimal places. It is also possible to protect any part of the sheet against accidental alteration. Negative numbers can be indicated by either a minus sign or brackets, and the /PRINT command allows control codes and text to be sent directly to a printer prior to printing all or part of a sheet.

An ingenious feature is the /HOLD command which permits any row or column to be held at a fixed position on the screen whilst the remainder of the sheet is scrolled. Thus, for example, top and left-hand side headings can be fixed permanently; further, row and column totals can be fixed at the right-hand side and bottom of



the screen if required, leaving a central window through which the whole sheet can be viewed and changed. This allows the effects of changes to be seen instantly. Although this feature is not quite as versatile as the Viewsheet windows, it is much easier and quicker to use and more than adequate for most purposes.

Although I began using Inter-sheet with the feeling that it was inferior to the other spreadsheet programs, it quickly won me over in use. The features of Inter-sheet, Viewsheet and Ultracalc all differ to a certain extent, and there is no overall best buy. It depends on what you need.

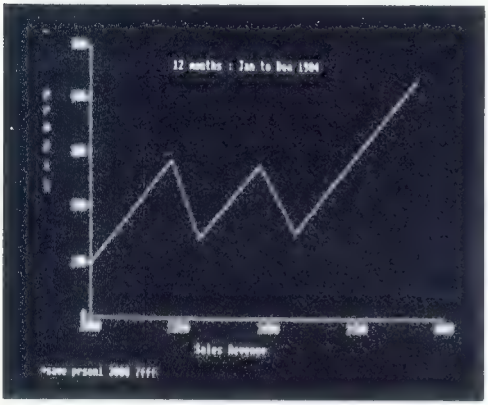
Viewsheet has sophisticated screen and print windows, but requires all columns to be set to the same width; Ultracalc works with the second processor but it is slow and not well-protected against errors; Inter-sheet has the 105 character screen and is very user-friendly. It also appears able to deal with a larger spreadsheet than either of its competitors and re-calculates more quickly. For the new user, I think Inter-sheet is the easiest system to begin with except for the fact that it is uncharacteristically let down by its manuals. The Introductory manual is very uneven and fails to get to grips with explaining why you might want to be using a spreadsheet, and how to make it do what you require. The Reference manual is adequate, but aimed at the competent user. A useful prompt card is provided. With this one reservation, I am happy to recommend Inter-sheet as a competent spreadsheet package.

However, I have not yet mentioned the major novel feature of Inter-sheet which is that it is part of Computer Concept's ROM-LINK system. The sheet program itself occupies one 16K ROM whereas the second ROM holds the ROM-LINK commands. These permit several (up to 16) sheets to be held in memory simultaneously in different segments of memory. Not only that, but information can be easily exchanged from one sheet to another, allowing the results of one set of calculations to be incorporated as data in another sheet, or even from one package to another. Other ROM-LINK packages allow sheets generated by Inter-sheet to be incorporated in documents, to be printed as graphs or charts, or to be fed into a database. This all happens directly in memory without first spooling the results to disc to be loaded again into another package. At present only Inter-chart (the graph and chart drawing package is available to fulfill this potential.

INTER-CHART

Inter-chart provides a means of

generating graphs and charts from data. It can operate in a stand-alone fashion, obtaining its data by keyboard entry or from an ASCII file, or it can import data from a spreadsheet held concurrently in memory. Thus data held in Inter-sheet can be displayed as a graph by instructing Inter-chart to access the relevant row or column. If the data consists of a set of labels (e.g. headings such as months of the year) and a set of numbers, it may be displayed in any of three formats (line graph, bar chart or pie chart). The type of chart can be quickly changed, as can the display mode (modes 0, 1, 2, 4 and 5 being possible).



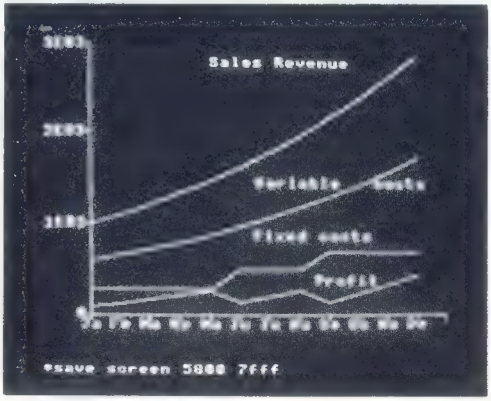
Computer Concept's usual form of menu allows the graph to be displayed and edited, and also permits the colours for each section of the chart to be selected from those available in a given mode. Further, a variety of hatched patterns can be obtained, although a considerable amount of experimenting is necessary to obtain the best results. Up to 16 different graphs can be held in one set of charts, and up to 16 sets of charts and sheets can be held simultaneously.

Further, line graphs and bar charts can be overlaid on one another (see screen 1), and headings and legends added, permitting a variety of presentations to be constructed. Unfortunately the display in some modes (particularly modes 2 and 5) cuts off some text at the edges. Perhaps the most generally useful display mode is mode 1 which allows four colours with 40 column text; for printout mode 0 gives the best detail. Both modes 0 and 1 use 20k of memory, limiting other data storage.

The editing features are impressive. Scales are initially adjusted automatically to fit the incoming data, but can then be adjusted manually, with logarithmic scales being available if required. Data points can optionally be marked with a cross or joined by a line. Labels can be displayed or omitted, and a single data point can be highlighted. Finally, when the screen display is satisfactory, a single press of a function key dumps the screen to your printer (Epson-compatible only). This is a fixed size (approx. 6" x 8"), although instructions are given for using other screen dump systems, such as Printmaster or Dumpmaster, which allow the size to be selected.

One odd feature is that data that consists of pairs of numbers (as when drawing an X-Y graph) must reside in adjacent columns of the spreadsheet. This is a pity as most spreadsheet models naturally keep such data in rows. Also, although headings are easy to add, they can be quite difficult to edit or remove.

Overall, ROM-LINK appears to be an excellent system, which works well in so far as I could test it. I was able to keep several different sheets in memory and to transfer information between them and the graphics package without problem. However, as even a small sheet takes up a fair deal of memory (e.g. a 15 column by 27 row budget took up 16% of the available mode 7 memory and 42% of the mode 3 memory), and as word processors are equally memory hungry, I am not convinced of the practical usefulness of this system on an unexpanded 32K Beeb. Once graphs or charts are constructed, memory require-



ments escalate rapidly. The ROM-LINK system provides a strong motivation for up-grading to a B-Plus or fitting a screen memory board (see our Retail price list for details).

The two packags reviewed here are quality products well up to the standards we have come to expect of Computer Concepts. Inter-sheet is easy to use and difficult to make disastrous mistakes with, although it is a pity that the introductory manual lets it down. Inter-chart is a valuable addition and, when the promised ROM-LINK wordprocessor and database arrive, they will all form an impressive system. Inter-sheet and Inter-chart are of a similar standard to software that is sold for 16-bit machines at considerably higher prices and, within the constraints of the BBC's limited memory, enables highly professional work to be carried out.

SPECIAL OFFER

We have extended our special offer on any orders for Inter-sheet and Inter-chart until 15th February 1986. Prices are as follows:

Inter-sheet and Inter-chart £80.25
Inter-sheet £52.35
Inter-chart £29.90

All prices include VAT and p & p. Overseas, please send same amount in Sterling - this covers extra p&p but not VAT). Send to "ROM Offer", P.O.Box 50, St Albans, Herts AL1 3YS.

POINTS ARISING POINTS ARISING POINTS ARISING POINTS

ROULETTE (BEEBUG Vol.4 No.6)

The move down routine included in the text with this program contained a misprint. The correct version of this routine is as follows:

```
1 IF PA.<&E01 THEN 10
2 *K.0 *T.|MF.A%=0TO(TOP-PA.)S.4
:A%!&E00=A%!PA.:N.|MPA.=&E00|M
0.|MDEL.0,4|MRUN|M
3 *FX138,0,128
4 END
```

The correction involves inserting a full stop (.) and changing one occurrence of | to ! both in line 2. This routine can be added to any Basic program that needs to be moved down in memory.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS

25 COLUMN MODE 2 - Nicholas Ahmon

Twenty five characters can be packed onto each line of a mode 2 screen by reducing the width of the characters. This reduces their readability a little but produces text a little like so-called 'computer writing'. The trick is to alter values in the VDU workspace:

```
MODE 2
?&34F=24  :REM No. bytes per character
?&30A=25  :REM Text window right hand limit
```

Changing the mode resets all to normal.

SINGLE KEY VERIFY - Dylan Reisenberger

This short function key definition will prompt for a filename and then check the contents of that file against the Basic program in memory, reporting any differences found.

```
*KEY0INPUTN$:CH=OPENIN(N$):E%=TRUE:FORQ%=PAGE TO TOP-1:P.~Q%:VDU11:IF?Q%<>BGET#CH T
HENE%=FALSE:P."Error at &";~Q%:N.:CLOSE#CH ELSE N.:P.'"All OK":CLOSE#CH|M
```

FUZZY COMMANDS

If you are frequently plagued by mis-typings, then Alan Webster's short program will be a boon. Now you can enter 'fuzzy' commands and the Beeb will still understand them.

Modern microcomputers are very dumb when it comes to communicating with the user. A simple typing mistake and the computer refuses to understand you. If you typed 'LOAD "name', forgetting to type the second quote, the BBC micro is smart enough to tell you that the quote is missing, but why doesn't it ignore this fact, and load the file anyway?

One of the most frequent errors we have found at the BEEBUG offices also arises with the LOAD command. The micro's abbreviation for LOAD is 'LO.', and this is then followed by a filename. If you happen to be even a reasonable typist, you could find your finger still on the Shift key when the '.' is pressed, thus giving you the line 'LO>"name"'.

This program assembles a small piece of machine code to intercept the error vector at &202/&203 and to check for a mistake. If the line of text started with 'LO>' then this routine automatically retrieves the filename from this 'fuzzy' command and loads the file in.

This may seem no more than a bit of fun, but it does show how we can modify existing Basic commands, and add new commands. In the near future we shall be publishing an article and program which will show you how to add your own Basic commands.

Type the program in and save it to tape or disc, and then run it. If the machine code is set up properly, you can save the machine code alone by typing *SAVE fuzcom 900 +FF <Return> (replace the 900 with D00 if you are using a cassette system). Then, to initialise the fuzzy command, simply type */fuzcom. To test that this code works, type 'LO>"name"' where 'name' is the filename of any Basic program. In the same way, you could set up a whole library of useful fuzzy commands to help the finger weary programmer.

PROGRAM NOTES

There are two procedures in this program. PROCbasic tests to see if Basic I or Basic II is fitted and sets up a ROM routine address accordingly. PRPCbasic also tests for the current active filing system, and sets an offset to be used in positioning the machine code. PROCassemble is the main routine which assembles the machine code.

We have included here a short description of each part of the program.
Lines 1060 - 1140: Set up the error vector to point to our new routine. Store the old contents in &70/71.
Lines 1170 - 1230: Entry to our routine - test for error number 4 ('Mistake').
Lines 1260 - 1290: Error is not a mistake, so exit to the default error vector stored at &70/71.
Lines 1320 - 1450: Find Basic's internal text pointer and look at the current statement being executed. If the statement is equal to the text at line 1480 ('LO>') then it is our error, otherwise exit to the default error handler.
Lines 1490 - 1630: Find the filename to be loaded. This always follows the load command, and is enclosed in quotes.
Lines 1650 - 1700: Load the file into memory.
Lines 1710 - 1740: Pull unwanted data from the stack, such as the return address of the routine and the RTI information.
Line 1750: Jump to the Basic ROM's continuation address. This allows further commands to be recognised.

```
  10 REM Program Fuzzy Commands
  20 REM Version B0.1
  30 REM Author  Alan R Webster
  40 REM BEEBUG  Jan / Feb 1986
  50 REM Program subject to copyright
  60 :
 100 CLS
 110 PROCbasic
 120 PROCassemble
 130 P.'"Fuzzy command set up."'
 140 END
 150 :
1000 DEFPROCassemble
1010 FOR PASS=0 TO 2 STEP 2
1020 P%=&900+offset
```

```
1030 [OPT PASS
1040 :
1050 .start
1060 LDA&202
1070 STA&70
1080 LDA&203
1090 STA&71
1100 LDA#load MOD256
1110 STA&202
1120 LDA#load DIV256
1130 STA&203
1140 RTS
1150 :
1160 .load
1170 PHA
1180 TYA
1190 PHA
1200 LDY#0
1210 LDA(&FD),Y
1220 CMP#4
1230 BEQmist
1240 :
1250 .exit
1260 PLA
1270 TAY
1280 PLA
1290 JMP(&70)
1300 :
1310 .mist
1320 LDX#0
1330 LDY&A
1340 DEY
1350 .a1
1360 LDA(&B),Y
1370 INY
1380 CPX#3
1390 BEQours
1400 CMP#32
1410 BEQa1
1420 CMP text,X
1430 BNEexit
1440 INX
1450 JMPa1
1460 :
1470 .text
1480 OPT FNEQUS("LO>")
1490 .a3 LDA(&B),Y
1500 INY
1510 .ours
1520 CMP#32
1530 BEQa3
1540 CMP#34
1550 BNEexit
1560 LDX#0
1570 .a4 LDA(&B),Y
1580 INY
1590 CMP#34
1600 BEQ1d
1610 STA fname,X
1620 INX
1630 JMPa4
1640 :
1650 .1d LDA#13
1660 STA fname,X
1670 LDA#&FF
1680 LDX #block MOD 256
1690 LDY #block DIV 256
1700 JSR&FFDD
1710 PLA:PLA
1720 PLA:PLA
1730 PLA:PLA
1740 PLA
1750 JMPcontinue
1760 :
1770 .block
1780 OPT FNEQUW(fname)
1790 OPT FNEQUS(STRING$(16,
"*"))
1800 :
1810 .fname
1820 OPT FNEQUS(STRING$(16,
"*"))
1830 ]
1840 NEXT
1850 ENDPROC
1860 :
1870 DEFFNEQUS(A$)
1880 $P%=A$
1890 P%=P%+LEN(A$)
1900 =0
1910 :
1920 DEFFNEQUW(A%)
1930 ?P%=A%:P%?1=A% DIV 256
1940 P%=P%+2
1950 =0
1960 :
1970 DEFPROCbasic
1980 A%=(?&8015)-48
1990 IFA%=1 continue=&8B0C
2000 IFA%=2 continue=&8B9B
2010 IFA%<1 OR A%>2 PRINT"N
ot a suitable Basic":END
2020 A%=0:Y%=0
2030 A%=(USR&FFDA)AND&FF000
0 DIV &10000
2040 IFA%=0 END
2050 IFA%<3 offset=&400
2060 IFA%>2 offset=0
2070 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS

BUG IN Z80 BASIC TRIG - D.W. Harris

ACS and ASN (but not ATN) give incorrect results with negative arguments:

	Correct	Z80 Basic
ACS(-1.0)	3.14...	0
ACS(-0.5)	2.09...	-1.047...
ACS(0)	1.57...	1.57...
ACS(0.5)	1.47...	1.47...
ACS(1.0)	0	very small

MEMOPLAN AND COBOL PROBLEMS IN Z80 - T.Hubbard

If you use the editor in Memoplan to edit a Cobol program and then run it under the debugger, Animator, you will run into problems because Memoplan uses mode 3 whereas Cobol (and Animator) expects mode 0. The solution is to type Ctrl-V 0 <Return> before running the compiled program to change to mode 0.

Build your own Database Manager

BEEBUG Filer Part 3

We add the final enhancements to BEEBUG's own database program, include a useful sorting routine and a powerful search option.

We hope you have already started experimenting with the BEEBUG Filer that we have developed and described in the previous two issues. This month we add the remaining features, including an update routine, a sort, and a search facility. With all these extra features, our BEEBUG Filer should prove a really useful program that will have many practical uses.

All the additional instructions, and any existing lines that have to be changed as a consequence, are listed together at the end. We will give you detailed practical instructions later on how to update your own version of Filer.

THE NEW FEATURES

If you have already tried using the program built up and described in the previous two articles, you will probably have suffered the inconvenience of being unable to change or update an existing record in your file without first deleting the record and then adding a new record. Because of the way Filer works this can be particularly bothersome if you want to modify a print format file, a feature we introduced last month.

A good, flexible update function is the first of the improvements to be described this month. At the same time, the opportunity has also been taken to make the 'ADD' function just as flexible.

No database system is complete without some means of sorting your records and this is the second new feature to be added. Although a relatively simple sort algorithm has been used, it has been implemented efficiently and is appropriate to the likely uses of Filer. By sorting the file a second time it is also possible to sort a file on more than one field.

The third enhancement allows the user to select records from the file by means of a 'search string'. This then determines which records will subsequently be displayed or printed. The search facility is particularly powerful in Filer because of the way it capitalises on the Basic EVAL function. You will find that Filer allows searches that are much more complex and sophisticated than those in many commercial packages.

The extra features added to the BEEBUG Filer are rounded off with an 'extend' function. This will allow you to extend the length of an existing database if you did not allocate sufficient space initially. This may not be a feature that you will need very often, but could prove a lifesaver when you do.

This is the final part of the current series, but to assist you further we have prepared a set of notes giving some general advice to help you make the most of what we consider to be an excellent database program. For the more technically minded we have also included detailed program information (more on all this later).

UPDATING RECORDS

A new UPDATE command has been added to Filer. In use, the command should always be followed by a record number, that of the one you want to change. The record specified will be displayed on the screen and the cursor positioned at the beginning of the first field. Any character typed in from the keyboard will replace the character at the current cursor position, and the cursor moved one character to the right. The cursor left and right keys may be used to move the cursor as required without changing the displayed data.

To insert additional characters, the f0 function key has been programmed to insert a space (visible as a background dot), which can then be filled with the new character. In addition, function key f1 has been set up to delete the character at the cursor position (effectively the opposite of f0). The Delete key functions as normal, deleting the character to the

left of the cursor position.

With the use of the two cursor keys described, and f0 and f1, you will now be able to edit each field of a record in turn. Essentially, what you see on the screen is what will be stored in the file. Once a field has been edited, you can move onto the next field by pressing Return. Pressing Return alone will leave any field just as it is.

The update facility has been implemented by completely replacing the previous PROCadd and FNinput routines by two revised and expanded ones, PROCupdate and FNupdate1. The new routines are also used by the existing ADD command so that the full field editing capability is also there when you enter a record for the first time (also when creating a file).

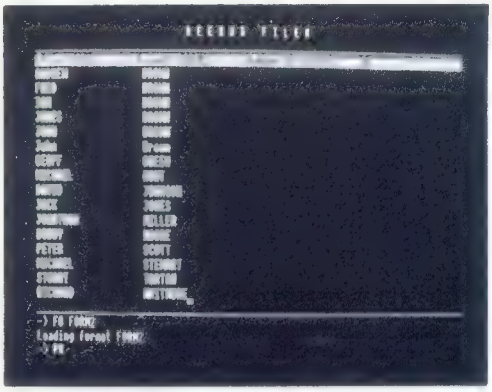
The ability to update records and edit fields should save a lot of retyping, particularly where very long fields are being used, or you have many fields within a record. In addition, by using the same routines for both initial entry and subsequent updating of records, very little additional memory is used up by the new update routines.

SORTING A DATABASE FILE

The BEEBUG Filer uses a 'Bubble Sort' to order the records in a file. This is based on the assumption that where order is important, you will wish to maintain a file in a particular order. Any new records are always appended to the end of a file and in these circumstances the Bubble Sort is a comparatively efficient way of moving the new records to their correct locations (see also the BEEBUG Workshop in Vol.4 No.7).

The obvious command, 'SORT', has been used to initiate sorting. This should be followed by the name of the field which is to be used as the key field in the sort (the file to be sorted should already be open). Remember that the BEEBUG Filer treats all information in files as being in the form of character strings, so a normal sort will be an alphabetic sort. However, there is provision to specify a numeric sort by adding '/N' immediately following the field name. For example, if a file contains two fields, one called NAME and one called AGE then:

```
SORT NAME
```



would sort the file alphabetically by NAME while the command:

```
SORT AGE/N
```

would sort the file numerically by AGE. In both cases the records in the file will be sorted in ascending order. There is no provision to sort in descending order, though the program could be readily modified to do this (see technical notes).

Although the sort routine is designed to sort the records in a file using only a single keyfield, it is still possible to sort on more than one key by performing two sorts in succession. For example, suppose the file referred to above had an additional field called FNAME (representing first name) then the two commands:

```
SORT FNAME
SORT NAME
```

would sort the records in the file in order of FNAME within NAME. Thus all the 'Browns' would be together but in order according to their first names.

Although the sort routine selected has been written as efficiently as possible, it will still take an appreciable amount of time to sort a file. There are ways by which this process can be semi-automated, described in the additional notes.

SELECTIVE RETRIEVAL

One of the most useful features of any database system is the ability to select which records should be displayed or printed by searching through a file for those records which match up to some specification. For example we might want to select those people whose surnames begin with the letter 'W', or who are over 20 years of age. Sometimes quite complex

searches are needed. We might in some file wish to select the records of all people with surnames between 'G' and 'S', who are between the ages of 21 and 40, and who live in the county of Kent. Given all the necessary information, Filer will cope even with something of that complexity.

The secret of Filer's power lies in its use of the EVAL function in BBC Basic (see this month's First Course for more information). This is a most useful language device not often encountered in other versions of Basic. EVAL really does all the work for us which means that very little additional code is added to the original program.

To use the search facility it is first necessary to use the SELECT command to specify the criteria to be used in selecting records. We will call this the 'search string'. The SELECT command thus takes the form:

SELECT <search string>

We will give you examples of actual search strings in a moment. Once a search string has been specified, the DISPLAY and PRINT commands will only retrieve records from the file that match up to the search string. However, you can always return to the normal situation (all records retrieved) by entering:

SELECT ALL (or SELECT all)

Remember that commands like SELECT may always be abbreviated to a minimum of two letters and may be in upper or lower case.

SPECIFYING A SEARCH STRING

A search string will usually refer to one or more field names. If we use the same kind of file, as an example, as we used in sorting, we can refer to fields NAME, FNAME and AGE. So we might say:

SELECT NAME>"J"

This would give us all the records where 'NAME' comes after the letter 'J' in the alphabet. We could easily make up a more complicated example:

SELECT NAME>"J" AND NAME<"S"

This would give us all the records with names between 'J' and 'S'.

When constructing a search string it is important to remember that within any Filer database, all information is treated and stored as a string of characters. Thus in the examples above the string constants are enclosed in quotes. If we wanted to perform a numerical comparison then we must convert from string to number format ourselves. For example:

SELECT VAL(AGE)>=21

This would select all records where the value of the 'AGE' field is greater than, or equal to, the number 21. In some ways it is like writing instructions in a version of Basic where the only variables are string variables.

Other Basic functions can be used as well, just as long as the search string can be evaluated to TRUE or FALSE (literally zero or non-zero). For example suppose we want to search for all records where the first name contains the letter 'R'. We could do this by writing:

SELECT INSTR(FNAME,"R")

The INSTR function (see the User Guide page 280) allows one string (given first) to be searched for one or more characters (given second). The result is either 0 (FALSE) if not found, or a value greater than 0 (TRUE) if it is.

Indeed, the search string can be any expression which can be evaluated by the EVAL function to give a value which can be interpreted as either TRUE or FALSE. The search string is evaluated for each record and only those records for which the result is TRUE are displayed or printed.

Because of this we can construct some novel if less useful search strings. How about this:

SELECT ASC(NAME)>GET

Now GET is a Basic function which will produce the ASCII value of the next character typed in at the keyboard. The ASC function will give us the ASCII value of the first character of NAME in each record. Thus whether a record is selected or not depends on how the first character of NAME compares with a key pressed on the keyboard AT THAT TIME. Thus the search can actually be made to change as the program works through the file.

The possibilities are almost endless, limited only by your own ingenuity. You are responsible, however, for making sure you get the syntax of any search string correct. Because this is evaluated by Basic, the BEEBUG Filer can only intercept any error message generated by Basic, display this on the screen and wait for your next command. Any data files will remain open and no serious damage is likely to occur. Watch out particularly

for 'No such variable'. You may well have made a mistake with a field name so that Filer cannot recognise it.

EXTENDING YOUR DATA FILE

With the best of intentions you may well find that you create a data file smaller than you eventually need. When this happens you can use the EXTEND command to extend your file. The format of the command is simply:

EXTEND <filename>

You will be asked to specify the new file size and the file will then be extended to its new length if this is possible. The file then remains open for further use.

Any error messages, such as 'Can't extend' or 'Disk full' will be reported and the command abandoned. The 'Can't extend' message will occur if the datafile is followed by another file on disc. Copying the file to another disc or removing the following file will usually allow the extension to take place. You will need to use the normal DFS commands for this file management.

FURTHER INFORMATION

To help you get the most out of the BEEBUG Filer, we have prepared a set of notes giving more technical information as well as a concise description of each command. It also includes a description of each function and procedure used, and a list of the principal variables. This is intended primarily for those with some programming knowledge who may wish to modify the published program. However, we have also included additional guidance on making the best use of many of the features of the BEEBUG Filer database.

The Filer notes are available on receipt of an A5 SAE. Please send this to our St Albans address marked 'BEEBUG Filer'. We will also consider publishing a further update to the BEEBUG Filer if there is sufficient interest and ideas to justify this.

We have also included the complete BEEBUG Filer on this month's magazine cassette/disc together with a datafile called NAMES with which you can practice using this month's new features.

ADDING THE NEW ROUTINES

Type in the new, or amended, instructions listed below and save them temporarily to disc using the *SPOOL command (see User Guide page 402). Then load into memory the combined parts 1 and 2 from the previous two articles, delete lines 3600 to 4320 (and line 5260) and append the new instructions using *EXEC. The resulting program can then be saved ready for use as FILER. Basic I users should ensure that OPENUP in line 3260 is replaced by OPENIN.

Remember, when typing in the additions, to avoid any unnecessary spaces, in order to conserve space. Although the complete program will run with PAGE at its default setting of &1900, you may find it better to lower PAGE to &1400 first (PAGE=&1400 <Return>). Don't lower PAGE below &1400 as buffer space is required here by the data files that BEEBUG Filer deals with.

```
1050 delete%=0:search$="-1"
1100 DATA 14
1170 DATA UPDATE,SORT,SELECT,EXTEND
1560 INPUT LINE"-> " command$
1660 IF C=4 THEN PROCupdate(pm$,0)
1730 IF C=11 THEN PROCupdate(pm$,1)
1740 IF C=12 THEN PROCsort(pm$)
1750 IF C=13 THEN PROCselect(pm$)
1760 IF C=14 THEN PROCextend(pm$)
2600 PRINTTAB(5,2)"Number of records:"
recn=VAL(FNupdate1("",24,2,4,"."))
2640 REPEAT:f=VAL(FNupdate1("",63,2,2,"
.")):UNTIL f<=maxf
2720 field$(I)=FNupdate1("",21,3+I,12,"
.")
2760 REPEAT:width%(I)=VAL(FNupdate1("",
56,3+I,2,".")):UNTIL width%(I)<=64
3260 F$=p$:F=OPENUP(F$):format%=0:searc
h$="-1"
4960 PROCread(I,0):IF EVAL(search$)=0 T
HEN 5000
4970 IF flag THEN PROCprint1(I) ELSE PR
OCdisplay1(I)
5240 PROChead(67,0,"Record: ",n,4)
6020 IF NOT open% ENDPROC
6180 CLOSE#F:open%=0:format%=0:PROCwind
ow1:CLS:PROCwindow2
6590 PROCwindow1:PROCinv(1):PRINTTAB(19
,0)"Format: ";p$:PROCinv(0):PROCwindow2
8000 DEF PROCupdate(p$,flag)
8020 LOCAL I,n:n=VAL(p$)
8040 IF NOT open% THEN PRINT"No file op
en":ENDPROC
8060 IF flag AND rec<2 THEN PRINT"No re
cords in file":ENDPROC
8080 IF flag AND (n<1 OR n>rec-1) THEN
PRINT"No such record":ENDPROC
8100 IF flag=0 AND rec>recn THEN PRINT"
File full":ENDPROC
```

```
 8120 PROCwindow1:PROCrecord:IF flag THE
N PROCread(n,0):PROCdisplay1(n)
 8140 IF flag=0 PROChead(67,0,"Record: "
,rec,4):n=rec
 8160 FOR I=1 TO f:IF flag ch$=record$(I
,0) ELSE ch$=""
 8180 record$(I,0)=FNupdate1(ch$,13,L*(I
-1)+1,width%(I),"."):NEXT I
 8200 PROCwindow2:IF FNask("Confirm (Y/N
): ")<3 PROCwrite(n,0):IF flag=0 rec=rec
+1:PTR#F=0:PRINT#F,rec
 8220 PROCwindow1:PROChead(58,0,"",rec-1
,4):PROCwindow2
 8240 ENDPROC
 8260 :
 8400 DEF FNupdate1(ch$,x,y,w,p$)
 8420 LOCALc,p:p=1:*FX4,1
 8440 IF ch$="" ch$=STRING$(w,p$)
 8460 VDU31,x,y:*FX225,140
 8480 REPEAT:c=GET
 8500 IF c=127 AND p>1 VDU8,ASCp$,8:ch$=
LEFT$(ch$,p-2)+p$+MID$(ch$,p):p=p-1
 8520 IF c=136 AND p>1 VDU8:p=p-1
 8540 IF c=137 AND p<w VDU9:p=p+1
 8560 IF c=140 AND p<=w ch$=LEFT$(LEFT$(
ch$,p-1)+p$+MID$(ch$,p),w):PRINTTAB(x,y)
ch$:VDU31,x+p-1,y
 8580 IF c=141 ch$=LEFT$(ch$,p-1)+MID$(c
h$,p+1)+p$:PRINTTAB(x,y)ch$:VDU31,x+p-1,
y
 8600 IF c>31 AND c<127 AND p<=w ch$=LEF
T$(ch$,p-1)+CHR$(c)+MID$(ch$,p+1):VDU c:
p=p+1
 8620 UNTIL c=13:*FX4,0
 8640 *FX225,0
 8660 =ch$
 8680 :
 8800 DEF PROCsort(p$)
 8820 LOCAL I,J,K,K1,R,flag,t$:I=2:t$=""
:format%=0
 8840 PROCwindow1:PROCinv(1):PRINTTAB(19
,0)STRING$(20," "):PROCinv(0):PROCwindow
2
 8860 IF NOT open% THEN PRINT"No file op
en":ENDPROC
 8880 IF rec<3 THEN PRINT"Nothing to sor
t":ENDPROC
 8900 IF p$="" THEN PRINT"No field given
":ENDPROC
 8920 J=INSTR(p$,"/"):IF J t$=MID$(p$,J+
1,1):p$=LEFT$(p$,J-1)
 8940 IF t$="N" t$="VAL" ELSE IF t$<>""
PRINT"Wrong switch":ENDPROC
 8960 t$=t$+"(record$(R,K))<"+t$+"(recor
d$(R,K1))"
 8980 R=0:REPEAT:R=R+1:UNTIL p$=field$(R
) OR R>f
 9000 IF R>f PRINT"No such field":ENDPRO
C
 9020 PRINT"Sorting file ";F$;" on field
 ";p$;" - please wait"
 9040 REPEAT:K=0:flag=-1:PROCread(rec-1,
K)
 9060 FOR J=rec-1 TO I STEP -1:K1=(K+1)M
OD2:PROCread(J-1,K1)
 9080 IF EVAL(t$) THEN PROCwrite(J-1,K):
PROCwrite(J,K1):flag=0:ELSE K=(K+1)MOD2
 9100 NEXT J:I=I+1
 9120 UNTIL flag:ENDPROC
 9140 :
10000 DEF PROCselect(p$)
10020 LOCAL I,p
10040 IF p$="" PRINT"No selection given"
:ENDPROC
10060 IF p$="ALL" OR p$="all" search$="-
1":ENDPROC
10080 IF NOT open% PRINT"No file open":E
NDPROC
10100 FOR I=1 TO f
10120 REPEAT:p=INSTR(p$,field$(I))
10140 IF p p$=LEFT$(p$,p-1)+"FNstrip(rec
ord$("+STR$(I)+",0),"+CHR$34+"."+CHR$34+
")"+MID$(p$,p+LEN(field$(I)))
10160 UNTIL p=0
10180 NEXT I:search$=p$
10200 ENDPROC
10220 :
10400 DEF PROCextend(p$)
10420 LOCAL n
10440 PROCclose:PROCopen(p$):IF NOT open
% ENDPROC
10460 PROCwindow1
10480 PRINTTAB(5,2)"Current file size: "
;recn;" records"
10500 PRINTTAB(5,4)"New file size:":REPE
AT:n=VAL(FNupdate1("",20,4,4,".")):UNTIL
 n>recn
10520 PROCwindow2
10540 PTR#F=EXT#F:PTR#F=256+n*recs:PRINT
#F,".":recn=n
10560 PTR#F=0:PRINT#F,rec,recn:PROCheade
r
10580 PRINT"File extended to ";recn;" re
cords"
10600 ENDPROC
10620 :
20240 IF open% THEN PROCinv(1):PRINT STR
ING$(80," "):PRINTTAB(1,0)"File: "F$:PRO
Chead(39,0,"Number of records: ",rec-1,4
)
20400 DEF PROCwindow1:IF w=2 X=POS:Y=VPO
S
20420 VDU28,0,19,79,3:w=1:ENDPROC
20460 VDU31,X,Y:w=2:ENDPROC
31012 IF ERR>189 AND ERR<208 REPORT:PRIN
T:GOTO140
31015 IF ERL=4960 PROCheader:REPORT:PRIN
T" in SELECT":GOTO 140
```

DISC RECOVERY

Virtually all disc users have experienced the disastrous loss of a sorely needed file. If action is taken quickly, Trevor Pullen's disc recovery program restores peace and calm, as well as the missing file.

It is all too easy to lose a valuable program or other file by deleting it. A moment of error can cost you hours of work. Discs can become corrupted too, with equally disastrous results. This program will provide a much-needed solution to this problem. It can be used to rescue programs or files of any length up to the maximum allowed by the disc or the DFS.

UNDERLYING PRINCIPLES

A disc is divided into concentric tracks each track being divided into sectors (10 for single density, 16 or 18 for double density) configured to hold 256 bytes of data. When a file is saved it goes onto the disc as a series of 256 byte blocks stored sequentially. For the computer to retrieve the file, it must know at which track and sector the file starts and how long the file is. This information is stored in the catalogue in the first sector along with other information such as the start and execution addresses for each program file.

When a *DELETE command is given, the catalogue reference is removed but the file itself is left untouched. However, if the disc is subsequently compacted or another file is saved then the area where the 'lost' file is stored may well be overwritten and the file lost for ever. Thus, if a file is accidentally deleted make no further saves to the disc and under no circumstances compact it until the required file has been rescued.

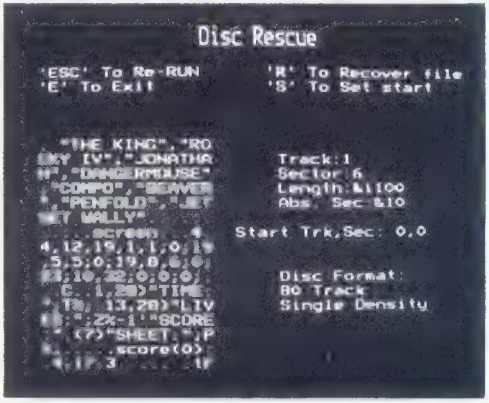
The computer will also lose track of the location and length of the files on the disc if the catalogue becomes corrupted. The catalogue is most prone to corruption because it is the most frequently accessed part of the disc - the rest of the disc may well still be okay. Once all the files have been rescued it is usually only necessary just to re-format the disc to make it re-usable.

So, to recover a 'lost' file we must manually locate its start and end. This information can then be used to patch the catalogue, or to allow the file to be 'lifted' off the disc and saved to a new one, automatically generating the required new catalogue entry. If rescue of a single deleted file is required then a fresh disc is not really necessary, as the original one can be used.

USING THE PROGRAM

Once you have typed in and saved the program, run it and, as prompted, specify the drive holding the program to be rescued. This will cause the program to display the contents of the first sector of that disc, along with the sector and track number and other information.

You should now scan through the disc, sector by sector, looking for your program. The '<' and '>' keys will display the previous and next sectors.



Alternatively, you can go straight to a particular sector by using the up and down cursor keys to increase and decrease the track number and the right and left cursor keys to increase and decrease the sector number, and then press the '<' or '>' key.

Only legal ASCII codes (32 to 126) are displayed, so identifying Basic programs can be quite difficult. The program will insert a solid white block in the first character position of the sector displayed

if it thinks you may be at the start of a Basic program. However, this is not an infallible guide. If you are not sure, you could recover the sectors anyway, and have a look at the file at a later stage.

Having located the start sector press 'S'. This will reset the displayed program length to &100 bytes (one block), and display the program start location on the disc. Now you need to find the end of the program or file. This is often the sector immediately before the beginning of the next program. Finally to rescue the program press 'R'. After a short pause, whilst the program is loaded into memory, a prompt for the new filename will be issued and you will be asked to insert the destination disc (of the same format as the source disc) into drive 0.

If the catalogue has been severely damaged this program will not be able to read the format of the disc and will probably print the 'Disc Error 18' message. In such a case, the program can be 'fooled' into reading the format, if a good disc of identical format is inserted into the drive and the program is restarted by pressing Escape. Once the first sector has been read, the corrupt disc can be reinserted and you can progress as before.

In the case of repeated disc error messages on all or most tracks, it is likely that the whole disc is severely corrupted and recovery of programs is probably impossible. You should be wary of continuing to use such a disc.

The moral here must be 'avoid problems before they happen if at all possible'. When developing programs use several discs and save updated copies frequently.

PROGRAM NOTES

PROCassemble: assembles a short routine which prints the ASCII contents of each sector as it is read in.

PROCinitialise: sets up the variables and flags for the rest of the program, identifies the format of the disc and displays the first sector on the screen.

PROCscan: controls scanning through the disc and handles 'DISC ERROR' reporting where necessary.

PROCrecover: recovers the target program or data file. The two variables 'Proglen' and 'Copylen' correspond to the actual length of the program or file to be copied and the length of the section currently being copied respectively.

If the file is longer than the available memory i.e. 73*256=&4900 bytes, 'Loop' will be set to 73. The file will be loaded and resaved in blocks of &4900 bytes until 'Copylen' is less than 73. 'Loop' is then set equal to 'Copylen' and the final section is saved.

PROCsavecat: saves the file or section of a file currently in memory, sets up the new catalogue information and reserves space on the new disc for the complete file using the OSFILE routine at &FFDD.

PROCsavemem: Once a catalogue entry has been generated, space reserved on the disc and the first section of the file saved, this procedure saves the rest of the file.

PROCforward/backward: responsible for moving through the correct number of sectors before changing to the next track.

PROCaccessdisc: general purpose disc access routine which will read or write the sector specified by 'Drv', 'Track', 'Sector' determined by the state of 'Read'. 'Start' is the location to which the sector will be read to, or saved from. The flag 'Recov' is used to inhibit the printing of each sector if the file is being 'recovered'.

PROCgetformat: sets up the basic variables 'Trk' and 'Sec', depending upon the format of the disc and initialises the DFS into the appropriate operating mode (single or double density).

PROCsetdrive: sets the current drive by executing a '*DRIVE n' command via the OSCLI routine at &FFF7.

```
10 REM Disc Crash Recovery
20 REM Version B2.1
30 REM Author: Trevor Pullen
40 REM Jan/Feb 1986
50 REM Subject to COPYRIGHT
60 :
100 ON ERROR GOTO 220
110 MODE 7
120 End=FALSE
130 PROCassemble
```

```
  140 REPEAT
  150 PROCinitialise
  160 UNTIL Format
  170 REPEAT
  180 PROCscan
  190 UNTIL Exit
  200 PROCrecover
  210 :
  220 IF End GOTO 270
  230 IF ERR=17 RUN
  240 IF ERR=204 error=TRUE:PROCsavecat
  250 IF ERR=190 OR ERR=197 OR ERR=198 O
R ERR=199 OR ERR=201 REPORT:PRINT;" Inse
rt a new disc."'"Hit any key to continue
";:X=GET:PROCsavecat
  260 REPORT:PRINT;" at line ";ERL
  270 *FX4,0
  280 VDU 23;11,255;0;0;0
  290 END
  300 :
 1000 DEFPROCassemble
 1010 DIMspace 30
 1020 FOR C=0 TO 2 STEP 2
 1030 P%=space
 1040 [
 1050 OPT C
 1060 .print
 1070 LDY #0
 1080 .again
 1090 LDA &3000,Y
 1100 CMP #32
 1110 BMI change
 1120 CMP #126
 1130 BPL change
 1140 JMP aclr      \Check range
 1150 .change
 1160 LDA #46       \Substitute for '.'
 1170 .aclr
 1180 JSR &FFE3     \Print data
 1190 INY
 1200 BNE again     \Get next chr.
 1210 RTS
 1220 ]
 1230 NEXT C
 1240 ENDPROC
 1250 :
 1260 DEFPROCinitialise
 1270 Exit=FALSE:Recov=FALSE
 1280 Read=TRUE:error=TRUE
 1290 Start=&3000
 1300 Len=1:Strk=0:Ssec=0:Sector=0
 1310 Track=0:CLS:*FX4,1
 1320 VDU 26;23;11,255;0;0;0
 1330 FOR Y=0 TO 1:PRINTTAB(12,Y);CHR$(1
41);"Disc Recovery":NEXT Y
 1340 VDU28,0,24,39,2,12
 1350 REPEAT INPUT''"Which drive (0,1,2,
3): "Sdrv:UNTIL Sdrv>=0 AND Sdrv<=3
 1360 Drv=Sdrv
 1370 PROCgetformat
 1380 IF Format=FALSE ENDPROC
 1390 CLS:VDU26;23;11,0;0;0;0
 1400 PROCaccessdisc
 1410 PRINTTAB(0,3)"'ESC' To Re-RUN";TAB
(0,4)"'E' To Exit";TAB(21,3)"'R' To Reco
ver file";TAB(21,4)"'S' To Set start"
 1420 PRINTTAB(22,9)"Track:0";TAB(22,10)
"Sector:0";TAB(22,11)"Length:&0";TAB(22,
12)"Abs. Sec:&0"
 1430 ENDPROC
 1440 :
 1450 DEFPROCscan
 1460 PRINTTAB(22,17)"Disc Format:"
 1470 IF Trk=80 PRINTTAB(22,18)"80 Track
" ELSE PRINTTAB(22,18)"40 Track"
 1480 IF Sec=17 PRINTTAB(22,19)"Double D
ensity" ELSE PRINTTAB(22,19)"Single Dens
ity"
 1490 PRINTTAB(18,14)"Start Trk,Sec: ";S
trk;",";Ssec;" "
 1500 PRINTTAB(30,11);~Len*256;"    "
 1510 REPEAT:A=GET:UNTIL A=139 OR A=138
OR A=137 OR A=136 OR A=44 OR A=46 OR A=6
0 OR A=62 OR A=83 OR A=115 OR A=82 OR A=
114 OR A=69 OR A=101
 1520 IF A=69 OR A=101 End=TRUE:CLS:GOTO
220
 1530 IF A=82 OR A=114 Exit=TRUE:ENDPROC
 1540 IF (A=83 OR A=115) Strk=Track:Ssec
=Sector:Len=1
 1550 *FX21,0
 1560 IF A=60 OR A=44 OR A=136 OR A=138
Dir$="B" ELSE Dir$="F"
 1570 IF A=83 OR A=115 ENDPROC
 1580 jump=TRUE
 1590 IF A>=136 jump=FALSE
 1600 IF Dir$="F" PROCforward
 1610 IF Dir$="B" PROCbackward
 1620 PRINTTAB(28,9);Track;" ";TAB(29,10
);Sector;" ";TAB(32,12);~(Track*(Sec+1)+
Sector);" "
 1630 IF jump PROCaccessdisc
 1640 IF ?&7A=0 PRINTTAB(2,6);SPC(37) EL
SE PRINTTAB(2,6);"Disc Error : &";~?&7A;
" "
 1650 ENDPROC
 1660 :
 1670 DEFPROCrecover
 1680 CLS
 1690 PRINTTAB(2,4)"Please wait...LOADING"
 1700 VDU28,0,24,39,2;23;11,255;0;0;0
 1710 Recov=TRUE
 1720 Track=Strk:Sector=Ssec
 1730 C%=-1
 1740 Proglen=Len:Copylen=Len
 1750 REPEAT
 1760 Read=TRUE
 1770 C%=C%+1
 1780 IF C%>0 AND Sdrv=0 PRINT'"Insert S
ource Disc. Hit any key";:X=GET
```

```
1790 Drv=Sdrv
1800 PROCsetdrive
1810 IF Copylen>73 Loop=73 ELSE Loop=Co
pylen
1820 IF C%>0 Track=Strk:Sector=Ssec
1830 FOR S%=0 TO Loop
1840 Start=&3000+256*S%
1850 PROCaccessdisc
1860 PROCforward
1870 NEXT S%
1880 Copylen=Copylen-Loop
1890 Strk=Track:Ssec=Sector
1900 IF C%=0 PROCsavecat ELSE PROCsavem
em
1910 UNTIL Copylen=0
1920 PRINT''"Do you wish to run again -
Y/N ";:X$=GET$
1930 IF X$="N" End=TRUE ELSE RUN
1940 ENDPROC
1950 :
1960 DEFPROCsavecat
1970 Drv=0
1980 PROCsetdrive
1990 CLS
2000 IF error PRINTTAB(2,4)"Bad Filenam
e - Press 'C' to continue":REPEAT X=GET:
UNTIL X=67
2010 error=FALSE
2020 CLS
2030 INPUTTAB(2,4)"Please enter new Fil
ename: "Fn$
2040 PRINT''"Insert Destination Disc int
o Drive 0.   Hit any key";:X=GET
2050 IF Sdrv<>0 PRINT'"Please wait...CO
PYING"
2060 $&A00=Fn$:!&70=&A00:!&72=PAGE
2070 !&7A=&3000
2080 !&7E=(&3000+256*Proglen)
2090 A%=0:X%=&70:Y%=0
2100 CALL &FFDD
2110 ENDPROC
2120 :
2130 DEFPROCsavemem
2140 IF Sdrv=0 PRINT'"Insert Destinatio
n Disc. Hit any key";:X=GET
2150 Track=0:Sector=1
2160 Drv=0:PROCsetdrive
2170 Start=&7A00:Read=TRUE
2180 PROCaccessdisc
2190 Ft=256*(?&7A0E AND &3)+?&7A0F+C%*7
4
2200 Track=Ft DIV (Sec+1):Sector=Ft MOD
(Sec+1)
2210 Read=FALSE
2220 FOR S%=0 TO Loop
2230 Start=&3000+S%*256
2240 PROCaccessdisc:PROCforward
2250 NEXT S%
2260 ENDPROC
2270 :
2280 DEFPROCforward
2290 IF Track=Trk-1 AND Sector=Sec ENDP
ROC
2300 IF A=139 Len=Len+Sec+1 ELSE Len=Le
n+1
2310 IF A=139 AND Track<Trk-1 Track=Tra
ck+1:ENDPROC
2320 Flag=FALSE
2330 IF Sector<Sec Sector=Sector+1 ELSE
Sector=0:Flag=TRUE
2340 IF Flag=TRUE AND Track<Trk-1 Track
=Track+1
2350 ENDPROC
2360 :
2370 DEFPROCbackward
2380 IF Track=0 AND Sector=0 ENDPROC
2390 IF A=138 AND Len>Sec+1 Len=Len-Sec
-1 ELSE IF Len>1 Len=Len-1
2400 IF A=138 AND Track>0 Track=Track-1
:ENDPROC
2410 Flag=FALSE
2420 IF Sector>0 Sector=Sector-1 ELSE S
ector=Sec:Flag=TRUE
2430  IF Flag=TRUE AND Track>0 Track=Tr
ack-1
2440 ENDPROC
2450 :
2460 DEFPROCaccessdisc
2470 ?&70=Drv:!&71=Start:?&75=3
2480 IF Read ?&76=&53 ELSE ?&76=&4B
2490 ?&77=Track:?&78=Sector:?&79=&21
2500 X%=&70:Y%=0:A%=&7F:CALL &FFF1
2510 IF Recov ENDPROC
2520 VDU28,0,24,15,8;30
2530 CALLprint:VDU 26
2540 IF ?&3000=13 AND (?&3001=0 OR ?&30
01=&FF) PRINTTAB(0,8);CHR$(255)
2550 ENDPROC
2560 :
2570 DEFPROCgetformat
2580 Drv=Sdrv
2590 PROCsetdrive
2600 X%=&85:Y%=0:A%=&7E
2610 CALL &FFF1:Trk=0
2620 IF !&85=&5A000 THEN Trk=80:Sec=17
2630 IF !&85=&32000 THEN Trk=80:Sec=9
2640 IF !&85=&30000 THEN Trk=40:Sec=17
2650 IF !&85=&19000 THEN Trk=40:Sec=9
2660 Format=TRUE
2670 IF Trk=0 PRINT"Unrecognised disc f
ormat":Format=FALSE
2680 IF Trk=0 PRINT'"Press any key to c
ontinue":X=GET
2690 ENDPROC
2700 :
2710 DEFPROCsetdrive
2720 $&A00="*DRIVE "+STR$(Drv)
2730 X%=0:Y%=&A:CALL&FFF7
2740 ENDPROC
```

VIEWSHEET AUTOBOOT

David Graham describes a utility for automating the use of Acornsoft's spreadsheet program, ViewSheet.

Many micro users who frequently use the same application, particularly word processing, spreadsheets and the like, find it helpful to configure discs of data so that all the initial settings and other custom features required can be automatically set up once Shift-Break has been pressed. The short program listed below enables the user to automatically select a datafile from a menu and then load this into Viewsheet (Acornsoft's popular spreadsheet package) ready for use.

The program also provides the opportunity to configure a printer. This is particularly useful for printing out large spreadsheets, because it allows you to select condensed print. On the Epson printer this gives a quite readable 120 columns on standard width paper.

The program should be tailored to your own requirements - the spreadsheets referred to in the listing are purely by way of example. When in operation, the program first asks if you wish to configure your printer. If you reply with a 'Y' then a set of control codes is sent to the printer. In the program, these are in line 140, and as listed send the codes to select condensed print on an Epson. For this feature to function correctly your printer should be on-line at this point. You can then proceed to select one of your spreadsheet files (containing Viewsheet data) which will be loaded into Viewsheet ready for further use.

To set the system up on your own Viewsheet data discs, type in the program as listed and tailor the details as described for your first disc. Additional menu selections can be added between lines 280 and 500, but remember to add corresponding procedure definitions, as in lines 1000 to 1080. Save the program to your Viewsheet data disc with a suitable name (e.g. STARTER). Now create a !BOOT file using the command:

```
*BUILD !BOOT
```

The one essential line to include is:

```
CHAIN"STARTER"
```

but other commands could also be included (see the Disc User Guide for more information). Use Escape to terminate entry of commands to the !BOOT file. Finally, type in:

```
*OPT4,3
```

Shift-Break will now automatically call the !BOOT file which will in turn call the starter program. The spreadsheet you want to work with can be selected and Viewsheet entered automatically.

If you have several discs of Viewsheet data, repeat the process for each one. You can save time by just copying the !BOOT file to each disc in turn. The menu program will need to be individually tailored for each disc.

```
  10 REM Program STARTER
  20 REM Version B1.0
  30 REM Author  David Graham
  40 REM BEEBUG  Jan/Feb 86
  50 REM Program subject to copyright
  60 :
 100 MODE7
 110 FOR I=1 TO 2:PRINTTAB(0,2+I)CHR$13
2;CHR$157;TAB(10)CHR$141;CHR$134;"VIEWSH
EET MENU":NEXT I
 120 PRINT''CHR$134;"Configure Printer?
 ";
 130 P$=GET$
 140 IF P$="Y" VDU 2,1,15,3
 150 PRINT''CHR$134;"1. Golf Club"
 160 PRINT'CHR$134;"2. Holiday"
 170 PRINT'CHR$134;"3. Bank Account"
 180 PRINT''CHR$130;"Select Option  ";
 190 A=GET-48
 200 IF A=1 THEN PROCone
 210 IF A=2 THEN PROCtwo
 220 IF A=3 THEN PROCthree
 500 *FX138,0,128
 510 END
 520 :
1000 DEFPROCone
1010 *KEY0 *SHEET|M MO.3|MLOAD V.GOLF|M
1020 ENDPROC
1030 DEFPROCtwo
1040 *KEY0 *SHEET|M MO.3|MLOAD V.HOLS|M
1050 ENDPROC
1060 DEFPROCthree
1070 *KEY0 *SHEET|M MO.3|MLOAD V.BANK|M
1080 ENDPROC
```

1st course

Using the EVAL Function

The EVAL function is a rare feature in most versions of Basic. It also has much hidden potential as Mike Williams reveals.

If you have read carefully through your User Guide when learning to program in BBC Basic you will have met the keyword EVAL. If you read the explanation therein, its function may have seemed a little odd, and probably not very useful. In fact, EVAL has considerable potential if you take the trouble to understand it fully. It can get you out of some otherwise tight spots, and will often reduce the number of instructions you need in a program.

PLOTTING GRAPHS

Very often, when writing a program, you will likely include a formula or equation of some kind. For example:

```
Y=10*SIN(X)
```

The part of the formula following the equals sign is called, in mathematics, an expression. In a sense, this expression is no more than a string of characters to which both we and Basic give a mathematical meaning. We could, however, treat it as a string of characters, and then using the EVAL function write the above instruction as follows:

```
formula$="10*SIN(X)"
Y=EVAL(formula$)
```

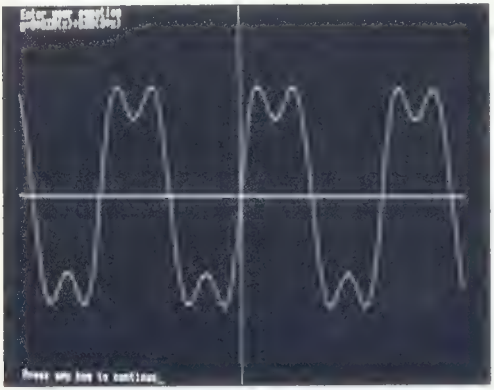
The EVAL function will evaluate any formula or expression that is written as a string of characters.

Now you may question the value of this, but we can readily demonstrate how useful this can be with an example. Suppose we want to write a program to draw graphs of mathematical functions on the screen. Being good programmers, the obvious way forward is to write a procedure to do just that, with one of the procedure's parameters being the equation to be plotted. However, the only way we can do this is by writing the equation as a character string, and immediately the role of EVAL becomes clear.

Not only that, but allowing the equation to be written as a character string means that it can be typed in when the program is running in response to a suitable prompt. Thus we can write a program that allows any mathematical equation (within reason) to be entered from the keyboard and then plotted on the screen.

```
  10 REM EVAL1
 100 MODE 0
 110 REPEAT
 120 CLS
 130 PRINT"Enter your equation"
 140 INPUT"y=" E$
 150 PROCaxes
 160 PROCgraph(E$)
 170 PRINTTAB(0,31)"Press any key to co
ntinue";:G=GET
 180 UNTIL FALSE
 190 END
 200 :
1000 DEF PROCaxes
1010 VDU29,640;512;
1020 MOVE0,1024:DRAW0,-1024
1030 MOVE-1280,0:DRAW1280,0
1040 ENDPROC
1050 :
1060 DEF PROCgraph(exp$)
1070 MOVE-1000,0
1080 FOR x=-10 TO 10 STEP 0.1
1090 y=EVAL(exp$)
1100 DRAW 64*x,102.4*y
1110 NEXT x
1120 ENDPROC
```

The program EVAL1 listed here shows you how this can be done. Two procedures are included. One, PROCaxes draws vertical and horizontal axes on the screen with the origin set to the central position (640, 512). The other procedure, PROCgraph, plots any function passed as the parameter exp$. The procedure is designed to plot any graph from x=-10 to x=+10 in steps of 0.1. The values used in line 1100 simply scale the graph plotted to a convenient screen size (remember that the graphics screen is nominally 1280 points horizontally by 1024 points vertically).



The program allows an equation to be entered and then plots it on the screen. Two curves that work well are:

```
y=3*SIN(x)+SIN(3*x)
y=EXP(x/5)*SIN(5*x)
```

You might like to try others of your own.

There is one point to note here, if everything is to work correctly. The EVAL function should evaluate the equation as though it had been written directly in the program in the first place. Since the loop that does the plotting uses the variable 'x' (a 'small' x) so the equation when entered must also use a small 'x'. You should be able to replace EVAL(exp$) by whatever function you have entered and end up with a legitimate Basic statement.

It is well worth while spending some time with this program, using it and modifying it, so as to get a good understanding of how EVAL works before reading on.

LOGICAL EXPRESSIONS

The character string that is evaluated by the EVAL function can be any string which Basic could evaluate to a numeric value. If you have been following previous articles in our series for beginners (Vol.4 Nos.2 & 3) you will know that logical values are represented in the computer as numerical values (zero for FALSE and non-zero for TRUE). As a result, we can also use EVAL to evaluate logical expressions and use such evaluations in IF-THEN statements, amongst others. This is illustrated in the next example program, EVAL2.

```
  10 REM EVAL2
 100 MODE 7
 110 DIM number(100)
 120 FOR I=1 TO 100
 130 number(I)=RND(99)
 140 NEXT I
 150 :
 160 REPEAT
 170 CLS
 180 PRINT"Select which numbers"
 190 INPUT t$
 200 test$="number(I)"+t$
 210 :
 220 FOR I=1 TO 100
 230 IF EVAL(test$) THEN PRINT number(I
)
 240 NEXT I
 250 :
 260 PRINT"Press any key to continue";:
G=GET
 270 UNTIL FALSE
 280 END
```

This sets up an array of 100 random numbers in the range 1 to 99. The program then allows any simple logical statement to be entered. This is then evaluated, using EVAL, to determine which of the 100 numbers is displayed on the screen. The program allows you to enter one of the logical operators (<, >, etc) followed by a number. So entering ">75" will display all the numbers which are greater than 75.

The technique used is simple, but an important one to understand. The test, entered in string form, is added to a further string to construct a complete test. Thus ">75", as above, is appended to the string "number(I)" in order to make the complete test string "number(I)>75". This is stored as test$ and evaluated by EVAL. Any such test will be either true or false, represented as 0 or -1.

Again, it is worth spending some time with this example. If you feel confident, you might like to consider how the program might be extended to handle more complex selection tests, for example picking out all the numbers between 25 and 75. We could write this directly with no trouble:

```
number(I)>25 AND number(I)<75
```

but it is a real tussle to organise in character form because of the essential repeated reference to the array. We will examine this further next month, and look at some more interesting applications of the EVAL function.

Title : Repton 2
Supplier : Superior Software
Price : £9.95 Cassette
: £11.95 Disc
Reviewer : Alan Webster
Rating : ****

Repton 2 is a follow up to the hugely successful Repton reviewed in Vol.4 No.4. Superior claim that Repton 2 is 'more than just a sequel, a new experience' and I must say that I agree with them.

The object is once again to roam the caverns in search of diamonds, but this time you must also collect pieces of a jigsaw puzzle in order to complete the game. Even the earth sections have to be collected, which means that in all you must collect 4744 earth sections, 1634 diamonds, 42 jigsaw pieces, kill all 18 monsters, use all of the 64 transporters, and collect the 'finish character'.

The game is certainly difficult, and

Title : Micro Olympics
Supplier : The Micro User
Price : £5.95 Cassette
: £7.95 Disc
Reviewer : Mike Williams
Rating : ***

From early and rather crude efforts, implementations of various sporting events have achieved a high level of realism, and a corresponding level of popularity in the charts. The Micro Olympics may not be the very best of its kind, but the graphics are good, and the events both varied and challenging.

There are races from 100 to 1500 metres where you pound the track by alternately pressing two keys of your choice as fast as possible. The shorter races are not too bad, but the longer events demand a level of stamina that my fingers were unable to supply. In the field events, such as the discus, hammer and high jump, you first demonstrate a

---

Title : Stairway to Hell
Supplier : Software Invasion
Price : £12.95 Cassette
: £15.95 Disc
Reviewer : Alan Webster
Rating : *****

Stairway to Hell is another game which is set in a series of caves and caverns and which has a competition offering £750 worth of computer equipment as first prize.

The game uses 4-colour mode 1 graphics and the 15 different screens are absolute masterpieces of graphical design. The colours are mixed well and the animation and use of characters is extremely impressive (although sound effects are very basic).

The object of the game is to wander or jump around collecting gold, trees, fruit and numerous other items. The game is very difficult, but you can pass over screens without even playing them. The scenario

---

Title : The Secret Diary of Adrian Mole
Supplier : Mosaic
Price : £9.95 Cassette
: £12.95 Disc
Reviewer : Mike Williams
Rating : ***

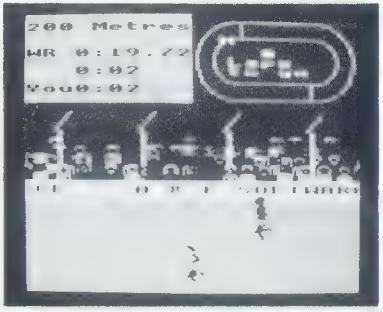
After the books and the recent TV series we now have the computer game. The computer version is mostly text with accompanying screen illustrations. Starting as a spotty schoolboy, you read through Adrian Mole's diary, making decisions which then determine the following story line. The aim is to make yourself (Adrian Mole) popular, not easy with his problems, a feminist girlfriend called Pandora (call me Box), a terror of an old age pensioner, and a perpetually lost dog. If that does not appeal, then you can, as the instructions suggest, set out to make youself very unpopular! Playing the part, I managed to lift my rating to 'suburban schoolboy', an achievement I was not too proud of.

would need a very long time to complete. As with Repton, you are confronted by a small section of the cavern and a funny looking creature which represents yourself. You can move the creature in one of four directions, and the whole screen scrolls smoothly as you do so.

A competition has been set for any who can complete Repton 2. You can win a T-shirt and £200. Anyone who completes Repton 2 certainly deserves the £200! If you fancy a tough challenge, then buy this game.

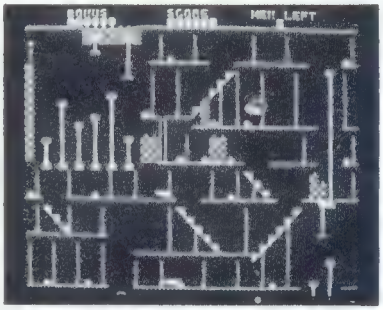
level of skill which then determines your subsequent success. This is complicated to follow initially, and rather artificial. The other events, javelin and long jump both depend on sufficient speed, as with running, to achieve a good result.

Overall, this game is good value and I would recommend it to anyone who enjoys such computer simulations. For those not yet hooked, there are probably more interesting ones available (for example 'Match Day' and 'Brian Jacks Super Stars'), but at a higher price.



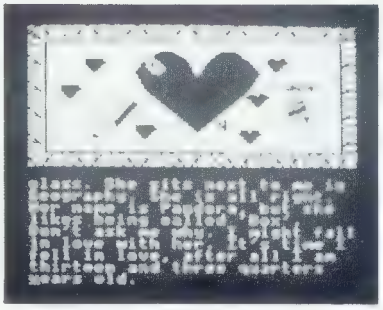
ranges from mines to blizzards, to the jungle, and finally to hell. The last three screens use some nice graphics to produce an orange-brown shade for the rocks and throughout the game the background is varied, colourful and well thought out.

The game is very difficult to play, and the prize of £750 should be safe for a while. If you like a good colourful game then this is certainly worth having, if only to see the quality of background graphics. Thoroughly recommended.



The programmers (Level 9) have done a good job, faithfully capturing most of the flavour of the original book. Those who are fans of Adrian Mole will enjoy this opportunity to play the part themselves.

However, once I had read right through the diary I felt little incentive to play again, particularly as the game is in four sections loaded in turn from cassette (in my case). Despite this, Adrian Mole is both popular and successful, and as a bit of Christmas fun the computer game will surely sell well.



BEEBUG WORKSHOP - Flexible Graphs

This month, Surac takes a different tack, by discussing a flexible procedure which will help in the drawing of graphs and diagrams.

The Beeb has splendid graphics but is not used as much as it might be for drawing graphs. The reason could be the difficulty of setting up the axes and their scales for any given graph.

If you know before you start what ranges to display along the axes it is not too difficult, though a little fiddly. However, if the program itself is generating the graph data and you don't know what's coming, things get a little hairy. The program can find out the ranges easily enough, but must then convert them into the right format.

This month we'll look at a way to draw and calibrate graph axes for any ranges, and then calculate scaling factors which fit the data on the screen. PROCgraph needs 4 input parameters, xlo and xhi, defining the upper and lower limits for the horizontal axis, and ylo and yhi doing the same vertically.

With this data, it draws the axes, together with 11 grid marks along each one. It then uses the scale range (lo-hi) for each axis to put a suitable value at each grid mark. The procedure also defines a graphics window between the axes, to prevent plotting across them, and sets up a small text window at the bottom of the screen for any messages you wish to display. Finally, it sets Xscale and Yscale to suitable values to fit your data onto the screen. Here's the procedure:

```
10000 DEF PROCgraph(xlo,xhi,ylo,yhi)
10010 VDU 26:CLG
10020 LOCAL mode,modeoffset,sclfact
10030 mode=?&355:IF mode>1 AND mode<>4
        THEN ENDPROC
10040 modeoffset=-(mode=0)
10050 Xscale=1040/(xhi-xlo)
10060 Yscale=800/(yhi-ylo)
10070 PROCaxes
10080 PROCgrads
10090 PROCwindows
10100 ENDPROC

11000 DEF PROCaxes
11010 LOCAL xl%,X%,Y%
11020 MOVE 160,200:DRAW 1200,200
11030 MOVE 160,196:DRAW 1200,196
11040 MOVE 156,200:DRAW 156,1000
11050 MOVE 158,200:DRAW 158,1000
11060 MOVE 160,200:DRAW 160,1000
11070 FOR X%=0 TO 1040 STEP 104
11080   xl%=160+X%
11090   MOVE xl%,180-(X% MOD 208)*.32:
          DRAW xl%,200
11100   NEXT
11110 FOR Y%=200 TO 1000 STEP 80
11120   MOVE 140,Y%:DRAW 160,Y%
11130   NEXT
11140 ENDPROC

11500 DEF PROCgrads
11510 LOCAL @%,scl$,X%,Y%
11520 @%=&01020509
11530 VDU 5
11540 sclfact=FNscale(xlo,xhi)
11550 FOR X%=0 TO 10
11560   scl$=FNval(xlo,xhi,X%)
11570   MOVE160+X%*104-LEN(scl$)*(16-8
          *modeoffset),
          172-(X%MOD2)*32
11580   PRINT scl$
11590   NEXT
11600 sclfact=FNscale(ylo,yhi)
11610 FOR Y%=0 TO 10
11620   MOVE modeoffset*70,Y%*80+212
11630   PRINT RIGHT$("    "
            +FNval(ylo,yhi,Y%),4)
11640   NEXT
11650 VDU 4
11660 ENDPROC
```

```
12000 DEF FNscale(lo,hi)
12010 LOCAL temp,sf
12020 temp=ABS(hi)
12030 IF ABS(lo)>temp THEN temp=ABS(lo)
12040 sf=LOG(temp) DIV 3
12050 IF temp<1 THEN sf=sf-1
12060 =10^(sf*3)

12500 DEF FNval(lo,hi,I%)
12510 LOCAL val,val$
12520 val=(lo+(hi-lo)*I%/10)/sclfact
12530 val$=LEFT$(STR$(val),4)
12540 IF RIGHT$(val$,1)="."THEN
        val$=LEFT$(val$,LEN(val$)-1)
12550 =val$

13000 DEF PROCwindows
13010 VDU 24,160;200;1279;1023;
13020 VDU 28,0,31,39+modeoffset*40,29
13030 VDU 29,160-xlo*Xscale;
          200-ylo*Yscale;
13040 ENDPROC
```

The procedure uses quite a number of handy tricks. Line 10030 looks at location &355, which (in OS 1.20) holds the current mode, to ensure that it is 0, 1 or 4. If not, the procedure exits. At line 10040, modeoffset is set to 1 for mode 0, otherwise it is zero. It is used later to position the scales correctly in 40 and 80 column modes.

Having calculated the scale factors for later graph drawing, PROCaxes draws the axes and the ticks along them. PROCgrads actually prints the scale marks. It has to handle an important feature of the program - it can take any range of numbers (e.g. 1, 0.001, 1000000), although there is only room for 4 characters for each graduation. The procedure uses FNscale to find an appropriate range for the scale numbers, by converting the numbers into engineering notation (e.g. milli or Mega).

For example, if an axis had to be graduated with the range 0.0-0.1, it would actually appear as 0-100; PROCgrads would convert to milli-units. Similarly, a range of 1000-1000000 would appear as .001-1.0 (Mega-units). When you use PROCgraph, you must remember that this apparent scale change can occur. The graduations are actually printed via FNval, which scales each value and then prints only the 4 most significant digits. It also suppresses (lines 12530 and 12540) any trailing decimal points.

This may seem quite a complicated way of marking the axes, but it is essential in a semi-intelligent program. It allows the routine both to handle any range and to show the scales with the maximum precision. At the same time, it does not cover the screen with confusing digits. PROCwindows sets the graphics window to the area between the axes. For ease of plotting, it also sets the graphics origin to the true origin of the graph, irrespective of where the axes are drawn.

Here is a short demo program, which you can merge with PROCgraph, to display the shape of any function:

```
  100 MODE4
  110 REPEAT
  120   INPUT "Formula? "form$
  130   INPUT "Range (lo,hi)? "rlo,rhi
  140   interval=(rhi-rlo)/50
  150   hival=-1E38
  160   loval=1E38
  170   PRINT "Finding upper and lower
limits"
  180   FOR X=rlo TO rhi STEP interval
  190     val=EVAL(form$)
  200     IF val>hival THEN hival=val
  210     IF val<loval THEN loval=val
  220     NEXT
  230   PROCgraph(rlo,rhi,
                  loval,hival)
  240   COLOUR 0:COLOUR 129:CLS
  250   MOVE rlo*Xscale,0
  260   FOR X=rlo TO rhi STEP interval
  270     DRAW X*Xscale,
               EVAL(form$)*Yscale
  280     NEXT
  290   UNTIL 0
  300 END
```

It calls for a formula - enter any valid Basic formula which uses only X as a variable. For example, you could try SIN(X)/X, or (X^2)*LOG(TAN(X)). Type it exactly as it would appear in a program. Then enter the high and low limits of X.

The program calculates the maximum and minimum values of the function in that range and then uses PROCgraph to plot suitable axes. Having set up the axes, the program draws the function; note how Xscale and Yscale convert true values to screen co-ordinates. You are then given a chance to enter another formula. A more elaborate demonstration version of this program appears on this month's magazine cassette/disc.

Writing your own compiler

(Part 2)

David Pilling continues his series on writing a compiler by looking at the task of syntax analysis.

In this part of the series, I will describe the syntax analysis section of the compiler. There is no doubt that this is the heart of any compiler and vital to all aspects of its operation. The syntax of a language is simply the set of rules which state what is a valid program and what is not. For instance, a Basic program must consist of a number of lines and each line must consist of a number of statements. There are then rules which say whether a statement is allowed or not. The syntax thus determines what the language actually is (e.g. Basic or Pascal).

Given a language to compile, a compact way of writing down its syntax is required. It would be no good to have to express the syntax in English. There are several well known techniques for doing this, such as the way the keyword DIV is described in the BBC User Guide:

<num-var> = <numeric> DIV <numeric>

This is a simplified form of the frequently used Backus-Naur formalism.

FLOW DIAGRAMS

An alternative way of describing syntax is by syntax graphs or flow diagrams. These have the advantage of being much easier to understand than the more mathematical methods. In Figure 1, the syntax graphs for arithmetic expressions are given. Note that these include both unary + and - operators (e.g. A = - B) which are handled by the bubble in the top left hand corner of the expression graph. The entire syntax of a language can be expressed this way.

Having decided what the syntax of a language is and how to represent it, what use is this in the construction of a compiler? At a trivial level, the answer is that a compiler has to read a program written in some language and needs to know if the program is correct. The syntax analysis part of the compiler performs this function of recognising correct programs. Acting like a filter, valid programs pass through it, while those with errors do not. There is another role for the syntax analyser which is providing a version of the program which can be easily translated into another language.

TREE REPRESENTATION

All programs, can be represented in the form of a tree. For example, consider:

A = 2 + B * C

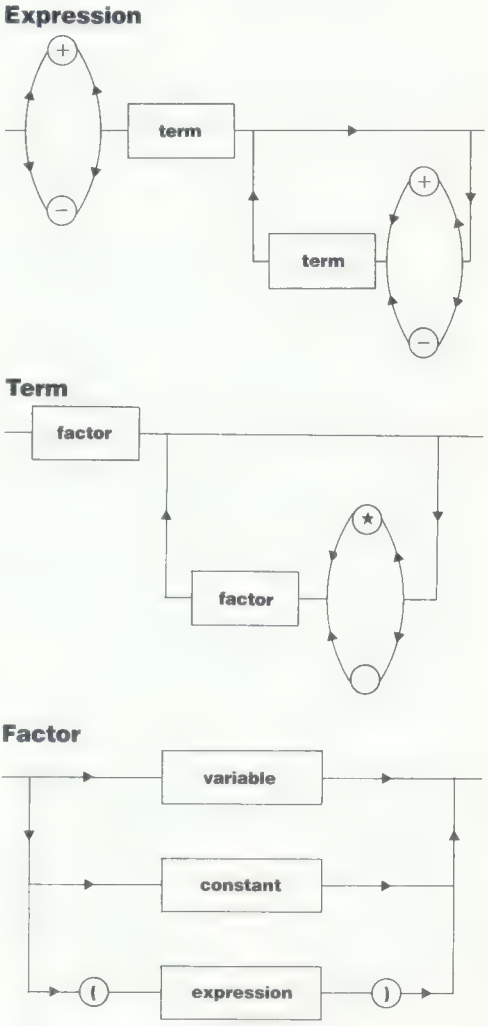


Figure one: syntax graphs for arithmetical expressions

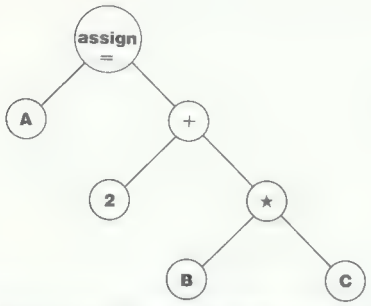


Figure two: Tree representation of the statement: A = 2 + B * C

This can be represented by the tree shown in Figure 2. Note that the branch nodes of the tree contain operators. Getting the program in this form, is crucial because once this has been done, it is comparatively simple to produce machine code for it. In some compilers, the syntax analysis section actually produces the tree data structure which is then passed to a separate code generation section. This has the advantage that the code generating part can have a detailed look at the branches of the tree and decide what to do.

An interesting thing about the tree representation is that by scanning it in different ways, we can easily produce different forms of the program. For example, a breadth first scan (in which all the nodes at one level are examined before moving on to the next) of Figure 2 would give:

```
(= A (+ 2 (* B C)))
```

This is just the statement in the prefix form used by languages like Lisp. Expressions can be written in three ways; infix where the operators appear between their operands as in everyday arithmetic, prefix where operators precede operands, and postfix, where operators appear after their operands. A depth first scan (that always tries to move down a level at each step) of Figure 2 gives:

```
A 2 B C * + =
```

which is the postfix form of the statement. This reverse Polish notation is used by the Forth language and by Hewlett-Packard calculators. As anyone who has struggled with either of these will confirm, it is much harder to use than infix notation. However, the important thing is that expressions in postfix notation can be easily executed by a stack based processor. The key to this simplicity is the absence of brackets in postfix expressions.

The 6502 processor has a number of registers, and operations can be carried out between these and each other, or with the contents of memory. On a stack based processor, all operations are between the stack and memory; we only have access to the top element of the stack. Typically, operations include adding the top two elements of the stack and replacing them with the result, or storing the top element of the stack in memory. Given a program in postfix form, execution of it is easy. All the processor has to do is move from left to right executing instructions and loading operands onto the stack.

Syntax graphs are important because they allow a very simple implementation of a syntax analyser or parser. If each diagram is represented by a procedure, and the various decision processes implemented, a syntax analyser for the language can be easily produced. As the program moves around the lines in the flow diagram, calls to the lexical analyser produce the next token from the source code.

THE COMPILER'S SYNTAX ANALYSER

The listing this month gives the syntax analysis part of the compiler. For the moment, focus on PROCsexp, PROCfactor and PROCterm. These are just the representation of the flow diagram Figure 1. They handle all the syntax analysis of expressions. By following this procedure from the flow diagrams for the complete language, the entire syntax analysis section of the compiler can be constructed.

An important thing to note is that these procedures are recursive; they keep calling each other. In technical terms, the method of syntax analysis used is recursive descent.

Obviously, a simple program recogniser, is not all we need. We are interested in getting hold of the tree form of the program. Thus the final interesting thing can be revealed. As the syntax analysis procedures follow the flow diagrams, they do a depth-first scan of the tree. By inserting calls of extra procedures in the correct places, it is possible to make the syntax analysis procedures produce either a postfix form of the program or the tree representation.

Such code generating procedures (usually denoted by having three capital letters) are liberally dotted throughout the syntax analyser. For the moment, imagine that these are operations for a stack-based processor. Assume that the expression looked at before is fed to the procedures PROCsexp, PROCterm and PROCfactor. Then the sequence of procedure calls given in Figure 3 will be produced. What the procedures do is as follows:

PROCLDL Load a literal value (i.e. a constant) onto the top of the stack.
PROCLDA Load the top of stack with a variable.
PROCMUL Multiply the top two stack elements.
PROCADD Add the top two stack elements.
PROCSTA Store the top of the stack in a variable.

It is clear that the sequence of instructions in Figure 3 will evaluate our expression. Note how the contents of the symbol table are used to get the correct addresses for variables which are referred to by their symbol table numbers.

```
PROCLDL(N)              ;N=2
PROCLDA(sa(ST%))        ;ST%=1
PROCLDA(sa(ST%))        ;ST%=2
PROCMUL                 ;
PROCADD                 ;
PROCSTA(sa(ST%))        ;ST%=0
```

n	st$(n)	sa(n)	stp(n)
0	"A"	100	id
1	"B"	95	id
2	"C"	90	id

symbol table

Figure three: The calls of code generating procedures produced for the expression A = 2 + B * C and the state of the symbol table.

Simply generalising the above process allows the syntax analyser to produce code for the entire language. Close examination of the listing will show this to be the case. There are, however, some points worth noting. First, look at PROCstate (line 2710). This procedure, is called to examine every statement in the source program. The most apparent feature is that it looks at the value of the first token in the statement and then calls the appropriate procedure to deal with it. One thus gets the huge string of IF's. This is a reflection of the fact that we only need to know the value of one token at any time to decide how the analysis of the program should go ahead.

Another important procedure is PROCexp. This analyses expressions. Every time something requires an expression as an argument, a call of this procedure is made. Look at line 3990 which is the procedure that handles the SOUND statement. Four calls of PROCexp deal with all its parameters. Being able to handle such a complicated thing as an expression with just a procedure call shows the power of this syntax analysis technique.

PROCexp, actually, is a higher level version of PROCsexp in that it not only handles simple expressions, by using PROCsexp, but also copes with comparatorial operators. The main use of these is in logical expressions like 'IF A>B THEN 100' and it brings us to the question of how such things work. The result of any logical expression is either true or false, and these are represented by the constant values 0 and -1. Operations like PROCGEQ are provided which compare the top two stack elements, and replace them with the appropriate logical value depending on their values. There is then a conditional jump operation given by PROCJMC which will jump if the top of stack element is false. Such a jump-if-false instruction fits into the code much more usefully than the more obvious jump-if-true that one might guess.

There are many other code generating procedures. The purpose of most of them should be guessable from their names. One of the nice things to do is to try and invent operations for the stack-based machine which are useful and will allow compact and fast programs to be written. This is an area which is great fun to experiment in.

It is important to realise what is going on here. The code generating procedures represent operations for a pseudo-machine. However, as will be shown in the next part, they actually generate 6502 code which carries out this function.

As stated, the syntax analysis procedures all call each other. However, there is one fundamental procedure called PROCprog (line 2520) from which the calls of all the others originate. This procedure does the syntax analysis for a complete program. A single call of it will cause an entire program to be analysed and have code generated for it. Looking at the listing in the first part of the series, you will find the call of PROCprog that

initiates compilation in line 1170.

The actual way the compiler works is thus as follows; a call of PROCprog unleashes a whole series of recursive calls of further syntax analysis procedures. As these go along, they get tokens from the source program by calling PROCLX and produce machine code by calling the code generating procedures. Thus all three processes, syntax analysis, lexical analysis and code generation are going on together and it is the syntax analyser which calls up the other two processes when it needs them.

Most real processors are not pure stack machines but instead have a large number of registers. For these, there are advantages in getting the tree representation of the program and then analysing this to find the optimum way of using the processor's registers.

More details on how to use the syntax graph method described here can be found in the exceptionally good book "Algoritms + Data Structures = Programs" by Nicklaus Wirth, Prentice Hall 1976.

Part 2 of the compiler listed this month should be appended to part 1 from the November issue (Vol.4 No.6). Part 3 will appear with the final part of this series, which will describe the code generating procedures and how to use the compiler.

```
 2490 REM Section #2
 2500 REM Syntax Analysis.
 2510 :
 2520 DEFPROCprog
 2530 IFT%=eop PROCEND:ENDPROC
 2540 PROCstate:IFT%=col ORT%=eoln PROCL
X:GOTO2530
 2550 ENDPROC
 2555 :
 2560 DEFPROCdef:LOCAL NP%,PS%,ST%
 2570 PROCEND
 2580 PROCLX:PROCC(proc):ST%=SN%:PROCLX
 2590 IFT%<>lb sa(ST%)=P%:GOTO2660
 2600 PS%=P%:PROCPS(1)
 2610 PROCLX:PROCC(id):PROCPP(sa(SN%))
 2620 PROCLX:NP%=NP%+1:IFT%=comma GOTO2610
 2630 PROCC(rb):PROCLX
 2640 V%=P%:P%=PS%:PROCPS(NP%):P%=V%
 2650 PROCPSX:sa(ST%)=P%:PROCJSR(PS%)
 2660 IFT%=eop ORT%=endproc GOTO2680
 2670 PROCstate:IFT%=col ORT%=eoln PROCL
X:GOTO2660
 2680 PROCC(endproc):PROCLX
 2690 IF NP%<>0 PROCPF(NP%,PS%)
 2700 PROCRTS:ENDPROC
 2705 :
 2710 DEFPROCstate
 2720 IFT%=eoln ORT%=col ORT%=else:GOTO2
920
 2730 IFT%=id PROCasign:GOTO2920
 2740 IFT%=proc PROCproc:GOTO2920
 2750 IFT%=rep PROCrep:GOTO2920
 2760 IFT%=whi PROCwhi:GOTO2920
 2770 IFT%=poke PROCpoke:GOTO2920
 2780 IFT%=gosub PROCgosub:GOTO2920
 2790 IFT%=return PROCreturn:GOTO2920
 2800 IFT%=inp PROCinp:GOTO2920
 2810 IFT%=vdu PROCvdu:GOTO2920
 2820 IFT%=if PROCif:GOTO2920
 2830 IFT%=for PROCfor:GOTO2920
 2840 IFT%=goto PROCgoto:GOTO2920
 2850 IFT%=print PROCprint:GOTO2920
 2860 IFT%=end PROCend:GOTO2920
 2870 IFT%=def PROCdef:GOTO2920
 2880 IFT%=sound PROCsound:GOTO2920
 2890 IFT%=move PROCmove:GOTO2920
 2900 IFT%=draw PROCdraw:GOTO2920
 2910 IFT%=cls PROCLX:PROCVDL(12):GOTO29
20
 2920 IF(T%<>eoln ANDT%<>col ANDT%<>else
) PROCERR
 2930 ENDPROC
 2940 :
 2950 DEFPROCend:PROCEND:PROCLX:ENDPROC
 2960 DEFPROCgosub:PROCLX:PROCC(lino):PR
OCJM(SN%):stp(SN%)=gosub:PROCLX:ENDPROC
 2970 DEFPROCreturn:PROCLX:PROCRTS:ENDPR
OC
 2980 DEFPROCpoke:PROCLX:PROCexp:PROCC(c
omma):PROCLX:PROCexp:PROCSTI:ENDPROC
 2990 DEFPROCasign:LOCALST%:ST%=SN%
 3000 PROCLX:PROCC(eq):PROCLX:PROCexp
 3010 PROCSTA(sa(ST%)):ENDPROC
 3015 :
 3020 DEFPROCproc:LOCAL ST%:ST%=SN%
 3030 PROCLX:IFT%<>lb GOTO3060
 3040 PROCLX:PROCexp:IFT%=comma GOTO3040
 3050 PROCC(rb):PROCLX
 3060 PROCJM(ST%):ENDPROC
 3065 :
 3070 DEFPROCvdu
 3080 PROCLX
 3090 PROCexp
 3100 IFT%=scol PROCVDD ELSEPROCVDU
 3110 IFT%=comma GOTO3080 ELSEIFT%=scol
PROCLX:IFFNEXP GOTO3090
 3120 ENDPROC
 3130 DEFPROCif:LOCAL L1%,L2%
 3140 PROCLX:PROCexp
 3150 L1%=P%:PROCJMC(0)
 3160 IFT%=then PROCLX:IFT%=lino PROCJM(
SN%):PROCLX:GOTO3180
```

```
 3170 PROCstate
 3180 IFT%=col:PROCLX:GOTO3170
 3190 IFT%<>else V%=P%:P%=L1%:PROCJMC(V%
):P%=V%:ENDPROC
 3200 L2%=P%:PROCJMP(0):V%=P%:P%=L1%:PRO
CJMC(V%):P%=V%
 3210 PROCLX
 3220 IFT%=lino PROCJM(SN%):PROCLX:GOTO3
240
 3230 PROCstate
 3240 IFT%=col:PROCLX:GOTO3230
 3250 V%=P%:P%=L2%:PROCJMP(V%):P%=V%:END
PROC
 3260 DEFPROCfor:LOCAL ST%,L1%:PROCLX
 3270 PROCC(id):ST%=SN%:PROCLX
 3280 PROCC(eq):PROCLX:PROCexp
 3290 PROCSTA(sa(ST%))
 3300 PROCC(to):PROCLX:PROCexp
 3310 IFT%=step PROCLX:PROCexp ELSEPROCL
DL(1)
 3320 PROCLX:L1%=P%
 3330 IFT%=eop ORT%=next GOTO3350
 3340 PROCstate:IFT%=col ORT%=eoln PROCL
X:GOTO3330
 3350 PROCC(next):PROCLX:PROCNXT(sa(ST%)
,L1%):ENDPROC
 3360 DEFPROCgoto:PROCLX:PROCC(lino):PRO
CJM(SN%):PROCLX:ENDPROC
 3370 :
 3380 DEFPROCprint:LOCALT1%
 3390 T1%=T%:PROCLX
 3400 IFT%=squote PROCVDL(13):PROCVDL(10
):GOTO3390
 3410 IFT%=scol GOTO3390
 3420 IFT%=string PROCSTR:GOTO3390
 3430 IFT%=comma PROCVDL(32):GOTO3390
 3440 IFT%=tab T1%=T%:PROCVDL(31):PROCLX
:PROCexp:PROCC(comma):PROCLX:PROCVDU:PRO
Cexp:PROCC(rb):PROCLX:PROCVDU:GOTO3400
 3450 IFFNEXP T1%=id:PROCexp:PROCPRI:GOT
O3400
 3460 IFT1%<>scol PROCVDL(13):PROCVDL(10
)
 3470 ENDPROC
 3480 DEFFNEXP:= T%=plus ORT%=minus OR T
%=lb ORT%=inkey ORT%=id ORT%=const ORT%=
sqr ORT%=sin ORT%=rnd ORT%=cos ORT%=pi O
RT%=fl ORT%=tr ORT%=peek
 3490 DEFPROCinp
 3500 Q=1
 3510 PROCLX
 3520 IFT%=squote PROCVDL(13):PROCVDL(10
):GOTO3500
 3530 IFT%=scol GOTO3510
 3540 IFT%=string PROCSTR:Q=0:GOTO3510
 3550 IFT%=comma Q=1:GOTO3510
 3560 IFT%=tab Q=0:PROCVDL(31):PROCLX:PR
OCexp:PROCC(comma):PROCLX:PROCVDU:PROCex
p:PROCC(rb):PROCLX:PROCVDU:GOTO3520
 3570 IFT%<>id ENDPROC
 3580 IFQ PROCVDL(63)
 3590 PROCINZ
 3600 PROCINP:PROCSTA(sa(SN%))
 3610 PROCLX:IFT%=comma GOTO3610
 3620 IFT%=id GOTO3600 ELSEIF FNINI:Q=1:
GOTO3520
 3630 ENDPROC
 3640 DEFFNINI:= T%=squote ORT%=scol ORT
%=string ORT%=tab
 3650 DEFPROCSTR
 3660 [JSRprs:]$P%=S$:P%=P%+LEN$+1
 3670 ENDPROC
 3680 DEFPROCexp:LOCAL T1%:PROCsexp
 3690 IFT%<>gt ANDT%<>lt ANDT%<>geq ANDT
%<>leq ANDT%<>eq ANDT%<>neq GOTO3720
 3700 T1%=T%:PROCLX:PROCsexp
 3710 IFT1%=gt PROCGTT ELSEIFT1%=lt PROC
LTT ELSEIFT1%=geq PROCGEQ ELSEIFT1%=leq
PROCLEQ ELSEIFT1%=eq PROCEQU ELSEIFT1%=n
eq PROCNEQ
 3720 ENDPROC
 3730 DEFPROCsexp
 3740 IFT%=plus PROCLX:PROCterm ELSEIFT%
=minus PROCLX:PROCterm:PROCNEG ELSE PROC
term
 3750 IFT%=plus PROCLX:PROCterm:PROCADD:
GOTO3750 ELSEIFT%=minus PROCLX:PROCterm:
PROCSUB:GOTO3750
 3760 ENDPROC
 3770 DEFPROCfactor:LOCAL ST%:ST%=SN%
 3780 IFT%=id PROCLX:PROCLDA(sa(ST%)):EN
DPROC
 3790 IFT%=const PROCLX:PROCLDL(N):ENDPR
OC
 3800 IFT%=lb PROCLX:PROCexp:PROCLX:ENDP
ROC
 3810 IFT%=inkey PROCLX:PROCterm:PROCINK
:ENDPROC
 3820 IFT%=peek PROCLX:PROCterm:PROCLDI:
ENDPROC
 3830 IFT%=sqr PROCLX:PROCterm:PROCSQR:E
NDPROC
 3840 IFT%=tr PROCLX:PROCLDL(-1):ENDPROC
 3850 IFT%=fl PROCLX:PROCLDL(0):ENDPROC
 3860 IFT%=sin PROCLX:PROCterm:PROCSIN:E
NDPROC
 3870 IFT%=cos PROCLX:PROCterm:PROCCOS:E
NDPROC
 3880 IFT%=pi PROCLX:PROCPI:ENDPROC
 3890 IFT%=rnd PROCLX:PROCC(lb):PROCLX:P
ROCexp:PROCC(rb):PROCLX:PROCRND:ENDPROC
 3900 PROCC(1)
 3910 ENDPROC
 3920 DEFPROCterm
 3930 PROCfactor
 3940 IFT%=times PROCLX:PROCfactor:PROCM
UL:GOTO3940 ELSEIFT%=slash PROCLX:PROCfa
ctor:PROCDIV:GOTO3940
 3950 ENDPROC
 3960 DEFPROCterm
```

49→

ADVENTURE GAMES ADVENTURE GAMES

by Mitch

Red Moon from Level 9 Computing, P.O. Box 39, Weston-super-Mare, Avon. Price £6.95 on cassette.

This game re-enacts a tale from a time when magic still worked and when mythical monsters guarded fabulous treasures. You have many weapons and magical spells to help you rescue the Red Moon crystal.

The game consists of over 200 locations of white text, but unfortunately there are no graphics in the version for the poor old Beeb. The commands appear to have gone back to the old two word format of the ATTACK DRAGON style and the dictionary of known words has shrunk. I trust this shrinkage is not a side effect of supplying pretty pictures to Uncle Clive's motley crew.

A feature of the game is the numerous fight sequences between yourself and various monsters, this is also in an old format of 'You strike the Vampire with a hit of 10 points - He strikes you with a hit of 5 points. You have 21 hit points left'. Casting spells by means of various artifacts and consumption of multi-coloured potions alternately decreases and increases your hit-point bank balance. This feature quickly becomes a pain in the posterior when the monsters, and their resurrected ghosts, continually reappear to block your progress to some interesting room.

Level 9 games have always been treated like the 'Emperor's New Clothes' in adventuring circles, with reviewers falling over their superlatives to admire them - there I've said it and I don't care! Trouble is, having waited a long time to say Level 9 has no clothes on, this game really has got a mass of puzzles and interesting objects. The comparatively low price tag means that this is a good buy and it would be churlish to say otherwise.

MAZES

Now to a subject which is the curse of the Adventuring Classes - the Maze. Mapping these twisty tunnels has always been the bane of my life until I was forced recently to adopt a more organised method in place of my usual mad thrashings. Instead of attempting to draw a conventional map, which I knew would very quickly resemble a plate of spaghetti, I decided to create a table.

The table should have a column for every location, which is identified by number and name of the object dropped there. Further columns for that entry should be made for every direction from the location. At starting location 1 we drop an object (e.g. BOOK) and insert this in the table location. We now move in the first empty direction column for the location (i.e. NORTH) and drop a further object (e.g. COIN). The move is recorded by inserting 2 in our NORTH column and entering the object in the next vacant location slot. The next move is again indicated by the first empty direction column for the current location and a repeat of the previous procedure. By this means we can systematically fill in the table which will achieve three main purposes.

(a) Any gap in the table presentation quickly shows a missed direction possibility.
(b) A short route between any points can be found and a conventional map drawn.
(c) Any special location will become self evident by its lack of occurrence in the table (i.e. It appears only once!). This last feature was a boon in the solving of 'Enthar Seven'.

LOCATION	DIRECTION									
	N	S	E	W	NE	SE	NW	SW	UP	DWN
1. BOOK	2	4	3							
2. COIN	3	1								
3. BUCKET	2									
4. SWORD	3									

The method is simple and obvious and it has only taken me three years to realize it!

HOME FINANCE MADE EASY

Using your micro to keep tabs on your bank account seems like a good idea. John Pitty and Mike Williams report on how three home finance packages compare.

Title : Bank
Supplier : Diamant Software,
7 Goodwood Avenue,
Manchester M23 9JQ.
Tel: 061-962 2708
Price : £12.50 Cassette or Disc
Reviewer : John Pitty

Title : Money Management
Supplier : Gemini Marketing Ltd.,
Gemini House, Concorde Road,
Exmouth, Devon EX8 4RS.
Tel: 0395-265165
Price : £12.95 Disc only
Reviewer : Mike Williams

Title : Personal Finance
Supplier : Kansas City Software,
Unit 3, Sutton Springs Wood,
Chesterfield.
Tel: 0246-850357
Price : £22.50 Disc
Reviewer : John Pitty

Many micro users, pondering the latest 'friendly' communique from their bank manager, have no doubt asked themselves questions such as: Do I really need software to manage my personal finances? Wouldn't pencil and paper serve just as well, and be cheaper? What's wrong with a bank statement? The answers to these questions depend not only on the quality of the software, but also on the facilities offered, and how well they can be controlled by the user. Whatever options are offered by a program, you must first determine whether they are what you require, do they work in the way you expect, and are they flexible enough for your needs.

BANK - Diamant Software

With this program your data is saved on disc, but it must still fit into memory for processing, limiting the number of transactions. To start a new account, details of your standing orders, which can also be deleted or amended, are entered via an unimpressive menu. This is the only menu in the whole program. If your standing order is not based on a yearly cycle, for instance rates payments over ten months, then for each of those ten months, the same details have to be typed in. A list of these entries is then sent to the printer, whereupon 'FINISHED (Y/N)' appears. Assuming this was asking if I had finished with this section I pressed 'Y', only to find, without warning, that I had quit the program! 'User friendly' seems to be in short supply these days!

On reloading the program, the standing order menu is again displayed, after which 'FINISHED(Y/N)' again appears. Pressing 'N' this time puts me in the transaction part of the program, a statement of which is untidily displayed on the screen. Two options only are offered now; 1) is a bank statement required?, or, 2) are fresh items to be entered? If neither option is required then you are in some difficulty as, incredibly, there is no other alternative. Also, no provision is made to allow amendment or deletion of transactions. Once entered, a statement is sent to screen and printer, plus a reminder of standing orders due for payment in the following week. No option is offered to return to a main menu. Again, the program just abruptly ends.

CONCLUSION

Prospective purchasers are likely to be very disappointed with this package. It is basic and unimaginative, and suffers from poor screen displays. No attempt has been made to structure the program round a main menu. Instead, it felt very muddled, with no real feeling of control.

MONEY MANAGEMENT - Gemini

This replaces the earlier 'Home Accounts' from Gemini, and is much more comprehensive as well as being only half the price. A 20 page manual is also included. The program runs in mode 7 with budgets, banking routines and printed reports selected via a main menu.

All headings can be customised to your

own requirements and the same program and files will handle up to 5 'asset' accounts (bank, building society, etc), and up to 5 'liability' accounts (credit cards, loans, etc). Details of all payments and receipts (including standing orders) can be entered and edited at will. You can also enter details from your bank (and other) statements to provide reconciliation of accounts. A 'budget' option allows you to set up budgets for all your accounting heads and to monitor progress, changing your budgets if you wish.

Budget Report at Jan 28 1985
DIRECT SPEND
Comparison of Expenditure and Budget

Subhead	Budget	Expended	Balance
Electricity	250.00	0.00	250.00
Oil	600.00	50.00	550.00
Telephone	150.00	0.00	150.00
Rates	600.00	55.98	544.02
Insurance	180.00	0.00	180.00
Household	3000.00	44.73	2955.27
House Maint'ce	1000.00	0.00	1000.00
Clothes	500.00	30.48	469.52
Garden	100.00	0.00	100.00
Presents	300.00	0.00	300.00
Motoring	500.00	16.94	483.06
Hols & Enttmt	2000.00	0.00	2000.00
Personal Self	500.00	0.00	500.00
Personal Spse	500.00	0.00	500.00
Miscellaneous	1000.00	0.00	1000.00
TOTALS	11180.00		10981.87
1 Mth budget=	931.67	198.13	

Example of MONEY MANAGEMENT Budgets

Gemini claim that one set of data files will be sufficient for a whole year's transactions, but all balances and other information can be carried forward at any time to a new set.

CONCLUSION

Most home users will find here all the accounting facilities they are likeky to need. The manual, though, is poorly produced and I for one found it difficult to follow. The program uses multi-coloured menu screens to excess, and with many (probably desirable) safeguards built in, proved cumbersome and time consuming in use. At the price, though, this must be a good buy.

PERSONAL FINANCE - Kansas City Systems

The 'User Guide' consists of 9 pages of computer listing paper formed into a booklet, giving a very comprehensive description of the many facilities. Transaction data can be written to a separate data disc, but if only one drive is available, you will be writing to the program disc, as this needs to be in place for continual access.

From the main menu, selecting an option chains to the relevant sub-menu for transactions, standing orders, checking bank statements and income/expenditure evaluation. The sixth and last option, Utilities, allows printing and sorting of transactions/standing orders, archiving transactions and creating, deleting or changing accounts.

Apart from two small points, all parts of the program are straightforward and easy to use. Each option is adequately covered by the handbook, although the 'getting started' section is a little confusing. When entering transactions, you are asked what category each item is allocated to. What isn't explained in 'getting started', is that when first starting an account, the first option to choose should be Income/Expense category evaluation, and, in what is a very useful facility, allocate a letter, A to Z, or number, Ø to 9 against selected items, e.g. H = household, C = car etc. With account information now available, selecting a start and end date, and a single or multiple category code, a printout of the transaction totals for each requested category can be made.

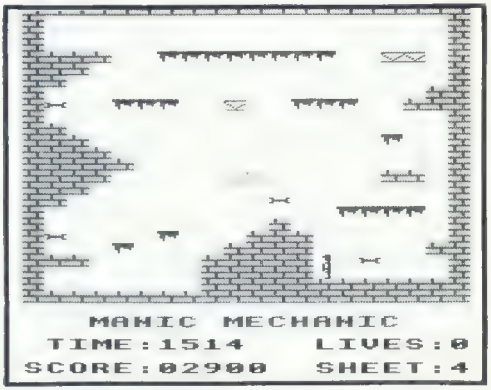
The second point to note is that of changing accounts. If a second account is created on the disc that you have your first account on, then the Income/Expense allocations (option 5), will be the same for both accounts. Changing the categories to suit the new account will affect those already designated for the first. Unless each account has the same categories, it would be advisable to use a separate disc.

CONCLUSION

This program offers a comprehensive range of facilities to the user, all based on easy to use menus, backed up by a genuine User Guide, certainly a program to be recommended, either for home, or even a club or association.

MANIC MECHANIC

The generator has broken down and the factory is slowly freezing up. Can Manic Mechanic collect all the spanners and reach the generator in time? Find out with this latest game from Jonathon Temple.



Your objective is to guide the manic mechanic through five different screens before reaching the generator, but it won't be easy. You'll have to contend with treacherous ice, malfunctioning conveyor belts and some dangerous jumps as you collect the five spanners on each screen. While you're doing this the time will be continually ticking away - if it reaches zero before you get to the generator, it's bye-bye mechanic.

The moving platforms also deserve a special mention. Instead of carrying you along, like any self-respecting conveyor belt, they'll gladly go off without you, and you will find yourself having to run to stay on them.

The keys to use are the usual 'Z' and 'X' for left and right and 'Return' to jump. In addition, you can pause in the game by pressing 'P', and continue with 'C'.

While temporarily stopped, 'Q' and 'S' will turn the sound on and off respectively; 'F' will finish a game, and 'R' will allow you to re-start a screen - useful if you become trapped, but you do lose a life.

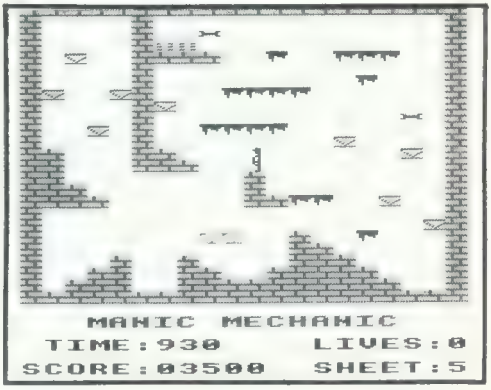
As the program is fairly lengthy, disc users (or users with PAGE set higher than &E00) should use a move-down routine such as the one below.

```
1 *K.0 F.A%=0TO(TOP-PA.)S.4:A%!&E00=
  A%!PA.:N.|MPA.=&E00|MO.|MG.100|M
2 *FX138,0,128
3 END
```

```
 10 REM PROGRAM MANIC MECHANIC
 20 REM VERSION B0.2
 30 REM AUTHOR  J. Temple
 40 REM BEEBUG  JAN/FEB 1986
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 ON ERROR GOTO 3560
110 MODE 7
120 PROCinit:PROCchars
130 PROCenvs:PROCtitle
140 REPEAT
150 MODE 2:PROCsetvars
160 REPEAT
170 PROCreset
180 REPEAT
190 PROCscreen
200 REPEAT
210 PROCman:PROClift
220 UNTIL E%
230 IF E%=1 PROCkilled
240 UNTIL Z%=0 OR E%>1
250 IF E%=2 PROCnext
260 UNTIL E%<>2
270 PROCend
280 MODE 7
290 PROChall
300 UNTIL FALSE
310 :
1000 DEFPROCman
1010 IFT%>0 T%=T%-1:VDU4:PRINTTAB(6,28)
;T%;"  ":VDU5 ELSE E%=1
1020 A%=X%:B%=Y%:C%=V%:D%=W%
1030 IFINKEY-56 PROCpause
1040 G%=POINT(X%+8,Y%-68):H%=POINT(X%+4
8,Y%-68)
1050 IFINKEY-74 IFJ%+F%=0 IFG%+H%>0 J%=
6:U%=(INKEY-98)-(INKEY-67):SOUND 18,2,10
,5
1060 IFJ% PROCjump:ENDPROC
1070 IFG%+H%=0 Y%=Y%-32:W%=W% EOR 1:F%=
F%+1:GOTO1120
```

```
 1080 IFF% SOUND 17,1,1,1:SOUND 18,0,0,0
:IFF%>3 E%=1 ELSE IFF% F%=FALSE
 1090 IFG%=8 OR H%=8 IFINKEY-98+INKEY-67
=0 X%=X%+(V%=225)*32-(V%=224)*32:W%=W% E
OR1:GOTO1120
 1100 IFINKEY-98 IFPOINT(X%-8,Y%)<>1 X%=
X%-32:Y%=Y%-(POINT(X%+24,Y%-32)=1)*32:W%
=W% EOR 1:SOUND 18,-10,50,1:IFV%<>225 V%
=225:W%=228
 1110 IFINKEY-67 IFPOINT(X%+72,Y%)<>1 X%
=X%+32:Y%=Y%-(POINT(X%+40,Y%-32)=1)*32:W
%=W% EOR 1:SOUND 18,-10,50,1:IFV%<>224 V
%=224:W%=226
 1120 IFD%<>W% GCOL3,15:MOVE A%,B%:VDUC%
,10,8,D%:MOVE X%,Y%:VDUV%,10,8,W%
 1130 IFPOINT(X%,Y%-20)=3 IFX%MOD64=0 IF
Y%MOD32=28 PROCspanner
 1140 ENDPROC
 1150 :
 1160 DEFPROCjump
 1170 X%=X%+A%(J%)*U%:Y%=Y%+B%(J%)
 1180 J%=J%-1:GCOL 3,15:MOVE A%,B%
 1190 VDUC%,10,8,D%:MOVE X%,Y%
 1200 VDUV%,10,8,W%
 1210 IFY%MOD32=28 IFPOINT(X%+8,Y%-68)+P
OINT(X%+48,Y%-68)>0 J%=0
 1220 IFPOINT(X%,Y%-20)=3 IFX%MOD64=0 IF
Y%MOD32=28 PROCspanner
 1230 IFU%=1 IFPOINT(X%+72,Y%-32)=1 U%=0
 1240 IFU%=-1 IFPOINT(X%-8,Y%-32)=1 U%=0
 1250 IFJ%=0 IFPOINT(X%+8,Y%-68)<1 IFPOI
NT(X%+48,Y%-68)<1 F%=1
 1260 ENDPROC
 1270 :
 1280 DEFPROCspanner
 1290 SOUND 17,4,100,1:K%=K%+1
 1300 IF K%=5 E%=2
 1310 L%=-1:REPEAT L%=L%+1
 1320 UNTIL K%(L%,1)*64=X% AND 1020-K%(L
%,2)*32=Y%
 1330 K%(L%,0)=TRUE:GCOL 3,3
 1340 MOVE X%,Y%:VDU230:PROCscore(100)
 1350 ENDPROC
 1360 :
 1370 DEFPROCpause
 1380 REPEAT N%=GET AND &DF
 1390 IF N%=81 THEN *FX 210,1
 1400 IF N%=83 THEN *FX 210,0
 1410 UNTIL N%=67 OR N%=70 OR N%=82
 1420 IF N%=70 E%=3 ELSE IF N%=82 E%=1
 1430 ENDPROC
 1440 :
 1450 DEFPROClift
 1460 GCOL3,5:MOVE Q%,R%:VDU232,232
 1470 Q%=Q%+IX%:MOVE Q%,R%:VDU232,232
 1480 IFQ%=L1% OR Q%=L2% IX%=-IX%
 1490 ENDPROC
 1500 :
 1510 DEFPROCscore(N%)
 1520 S%=S%+N%:VDU4,17,7,31,6,30
 1530 PRINT LEFT$("00000",5-LEN(STR$(S%)
))+STR$(S%)
 1540 VDU5
 1550 ENDPROC
 1560 :
 1570 DEFPROCkilled
 1580 Z%=Z%-1:SOUND 0,1,100,2
 1590 IF T%=0 VDU19,1,6;0;19,5,6;0;:Z%=0
 1600 TIME=0:REPEAT UNTIL TIME>100
 1610 ENDPROC
 1620 :
 1630 DEFPROCend
 1640 *FX 15,0
 1650 VDU4,28,4,13,15,11,12,26,5
 1660 PROCprint("GAME OVER",352,636,1,3)
 1670 TIME=0:REPEAT UNTIL TIME>200
 1680 ENDPROC
 1690 :
 1700 DEFPROChall
 1710 N%=-1:FOR L%=7 TO 0 STEP -1
 1720 IF S%>S%(L%) N%=L%
 1730 NEXT
 1740 IF N%>-1 PROCcongrats
 1750 PROCdisplay("Hall of Fame")
 1760 PRINTTAB(7,23);CHR$131"Press SPACE
 BAR to play";
 1770 REPEAT UNTIL GET=32
 1780 ENDPROC
 1790 :
 1800 DEFPROCnext
 1810 PROCscore(P%*250):P%=P%+1
 1820 IF P%=5 Z%=Z%+1
 1830 IF P%=6 PROCcomplete
 1840 TIME=0:REPEAT UNTIL TIME>200
 1850 ENDPROC
 1860 :
 1870 DEFPROCcomplete
 1880 PROCscore(Z%*500+T%):P%=1
 1890 SB%=SB%-250:T%=SB%:VDU19,8,0;0;
 1900 RESTORE 1960:N%=81:*FX 15,0
 1910 FOR L%=1 TO 10:READ A%,D%:N%=N%+A%
 1920 SOUND 1,-10,N%,D%
 1930 SOUND 2,-5,N%+48,D%:NEXT
 1940 ENDPROC
 1950 :
 1960 DATA 0,4,8,4,8,4,4,4,8,8,-12,8,4,8
,-12,8,8,8,-16,8
 1970 :
 1980 DEFPROCcongrats
 1990 LOCAL A%,L%,X%,Y%
 2000 FOR L%=6 TO N% STEP -1
 2010 S%(L%+1)=S%(L%):H$(L%+1)=H$(L%)
 2020 NEXT
 2030 S%(N%)=S%:H$(N%)=""
 2040 PROCdisplay("Congratulations!")
 2050 X%=&70:VDU 31,21,6+N%*2
 2060 !&70=&200C0A00:?&74=127:CALL &FFF1
 2070 H$(N%)=LEFT$($&A00,12)
 2080 ENDPROC
 2090 :
```

```
 2100 DEFPROCdisplay(M$)
 2110 VDU12,31,4,6,136
 2120 PROClarge(5,0,131,"SCORES")
 2130 FOR L%=3 TO 4
 2140 PRINTTAB((40-LEN(M$))/2-3,L%);CHR$
141;CHR$134;M$
 2150 NEXT
 2160 FOR L%=0 TO 7
 2170 PRINTTAB(5,6+L%*2);L%+1;" ... ";S%
(L%);TAB(17,6+L%*2);"... ";H$(L%)
 2180 NEXT
 2190 ENDPROC
 2200 :
 2210 DEFPROCprint(T$,X,Y,A,B)
 2220 GCOL 0,A:MOVE X,Y:PRINT T$
 2230 GCOL 0,B:MOVE X-8,Y-4:PRINT T$
 2240 ENDPROC
 2250 :
 2260 DEFPROCsetvars
 2270 Z%=3:S%=0:P%=1:SB%=2000:T%=SB%
 2280 FOR L%=9 TO 15:VDU 19,L%,7;0;
 2290 NEXT
 2300 ENDPROC
 2310 :
 2320 DEFPROCreset
 2330 FOR L%=0 TO 4:K%(L%,0)=FALSE
 2340 NEXT:K%=0
 2350 ENDPROC
 2360 :
 2370 DEFPROCinit
 2380 DIM W$(2),A%(6),B%(6),K%(4,2),S%(7
),H$(7)
 2390 C$=CHR$17
 2400 W$(0)=C$+CHR$1+C$+CHR$131+CHR$231
 2410 W$(1)=C$+CHR$6+C$+CHR$132+CHR$232
 2420 W$(2)=C$+CHR$8+C$+CHR$128+CHR$233
 2430 RESTORE2510
 2440 FOR L%=1 TO 6:READ A%(L%),B%(L%)
 2450 NEXT
 2460 FOR L%=0 TO 7
 2470 S%(L%)=4000-L%*500:READ H$(L%)
 2480 NEXT
 2490 ENDPROC
 2500 :
 2510 DATA 0,-32,32,-32,32,0,32,0,32,32,
0,32
 2520 DATA "THE KING","ROCKY IV","JONATH
AN","DANGERMOUSE","COMPO","BEAVER","PENF
OLD","JETSET WALLY"
 2530 :
 2540 DEFPROCscreen
 2550 VDU4,12,19,1,1;0;19,5,5;0;19,8,6;0
;23;10,32;0;0;0;
 2560 PRINTTAB(1,28)"TIME:";T%;TAB(13,28
)"LIVES:";Z%-1'"SCORE:"SPC(7)"SHEET:";P%
;
 2570 PROCscore(0):VDU4,17,3
 2580 RESTORE 2950:IFP%>1 PROCread
 2590 READ N%,K$,L1%,L2%,L3%,W$
 2600 FOR L%=0 TO 4
 2610 K%(L%,1)=ASC(MID$(K$,L%*2+1))-65
 2620 K%(L%,2)=ASC(MID$(K$,L%*2+2))-65
 2630 IFK%(L%,0)=FALSE VDU31,K%(L%,1),K%
(L%,2),230
 2640 NEXT
 2650 FOR L%=0 TO 24 STEP 24
 2660 PRINTTAB(0,L%) STRING$(20,W$(0))
 2670 NEXT
 2680 FOR L%=1 TO 23
 2690 PRINTTAB(0,L%) W$(0);TAB(19,L%) W$(0)
 2700 NEXT
 2710 FOR L%=1 TO N%*4-3 STEP 4
 2720 A$=MID$(W$,L%,4)
 2730 N1%=EVAL("&"+MID$(A$,3))
 2740 N2%=N1% AND31:N3%=N1% AND32
 2750 N4%=(N1% AND192)DIV64
 2760 D$="":IFN3% D$=CHR$(10)+CHR$(8)
 2770 PRINTTAB(ASC(A$)-65,ASC(MID$(A$,2)
)-65) STRING$(N2%-1,W$(N4%)+D$)+W$(N4%)
 2780 NEXT
 2790 IFP%=5 VDU17,4,17,128,31,6,2,234,2
35,10,8,8,17,136,236,236
```




```
 2800 Q%=L1%:R%=L3%:IX%=16
 2810 VDU17,7,17,128,5
 2820 GCOL3,5:MOVE Q%,R%:VDU232,232
 2830 E%=FALSE:F%=FALSE:J%=0
 2840 X%=128:Y%=956:V%=224:W%=226
 2850 GCOL3,15:MOVE X%,Y%:VDUV%,10,8,W%
 2860 PROCprint("MANIC MECHANIC",192,188
,4,6)
 2870 ENDPROC
 2880 :
 2890 DEFPROCread
 2900 FOR L%=1 TO P%-1
 2910 READ N%,K$,L1%,L2%,L3%,W$
 2920 NEXT
 2930 ENDPROC
 2940 :
 2950 DATA 25,SHCIPLCRSW,320,832,604
 2960 DATA BE03HE03ME44FF41SF01NH23FI02Q
I02GJ02KJ41DK01QJ03BL03BM03RM01BN04PN01S
P01PR42ES41HS25KS83CT81RU02BX10
```

```
 2970 DATA 32,QGSKCQCUSV,192,640,636
 2980 DATA CE0lFE82JE8lME86SG22RH0lCI4lF
I82JI8lMI08BL0lMM4lPM8lSM0lRP42DQ0lMQ84B
R0lDR02BS45IS0lHT03QT82OT0lHU04NU42CX0lH
V05FW08RW0lEX0AQX03
 2990 DATA 25,PCBGBMDTRS,384,896,668
 3000 DATA CE42GE84PE4lMF0lBI0lHI0CBJ02E
K8lQL4lSN0lBO0lOO8lRO02BP02FP82JP0lJQ03O
R42BU24PU83CV23HV82LU82DW22EX2l
 3010 DATA 27,FGBILQBTPV,416,800,572
 3020 DATA BE23CE22DE0lGE88QE42SH0lEI83J
I4lMI83RI02QL8lBK27CL25DM23EN0lQO02OR84L
S0lGT8lKT03EU8lJU04SU0lBV02IV05IW05IX0B
 3030 DATA 35,ICRJJUBVRV,256,640,412
 3040 DATA FB2DCE4lGE03LE8lOE83PG8lBH4lE
H4lJH84GI4lDK4lIK84OL4lBM25GM0lRM4lGN02K
O23CP22DQ0lLQ0lMQ82QQ4lSS4lMT0lPT8lMU02E
V23HV23MV03DW22IW0lLW05CX03IX09
 3050 :
 3060 DEFPROCchars
 3070 VDU23,224,48,56,48,48,32,48,40,40
 3080 VDU23,225,12,28,12,12,4,12,20,20
 3090 VDU23,226,40,56,40,48,32,32,32,48
 3100 VDU23,227,40,56,40,48,40,168,200,1
2
 3110 VDU23,228,20,28,20,12,4,4,4,12
 3120 VDU23,229,20,28,20,12,20,21,19,48
 3130 VDU23,230,0,0,195,126,126,195,0,0
 3140 VDU23,231,223,223,223,0,-5,-5,-5,0
 3150 VDU23,232,0,-1,128,65,34,20,8,-1
 3160 VDU23,233,-1,-1,127,110,98,32,32,0
 3170 VDU23,234,0,0,0,3,63,127,225,237
 3180 VDU23,235,0,0,0,228,242,-3,-4,-2
 3190 VDU23,236,-1,-1,153,-1,153,-1,153,
-1
 3200 ENDPROC
 3210 :
 3220 DEFPROCenvs
 3230 ENVELOPE 1,1,0,0,0,0,0,0,0,90,-1,-2,
-3,97,97
 3240 ENVELOPE 2,133,8,4,8,3,1,1,126,0,0
,-10,80,0
 3250 ENVELOPE 3,3,-1,-1,-1,70,50,1,100,
-1,-1,-10,101,5
 3260 ENVELOPE 4,133,8,4,8,3,1,1,126,0,0
,-10,126,0
 3270 ENDPROC
 3280 :
 3290 DEFPROCtitle
 3300 VDU 23;10,32;0;0;0;
 3310 FOR L%=1 TO 5
 3320 PROClarge(L%*7-6,1,129,MID$("MANIC
",L%,1))
 3330 NEXT
 3340 PROClarge(1,4,131,"MECHANIC")
 3350 PRINT'TAB(9);CHR$134;"by Jonathan
Temple"
 3360 TIME=0:REPEAT UNTIL TIME>500
 3370 ENDPROC
 3380 :
 3390 DEFPROClarge(V%,W%,C%,M$)
 3400 LOCAL A%,B%,L%,M%,S%,X%,Y%
 3410 FOR L%=1 TO LEN(M$)
 3420 ?&70=ASC(MID$(M$,L%)):?&79=0
 3430 X%=&70:Y%=0:A%=10:CALL &FFF1
 3440 FOR Y%=0 TO 6 STEP 3
 3450 VDU 31,V%,W%+Y%/3
 3460 IF L%=1 VDU C%+16,154
 3470 FOR X%=0 TO 6 STEP 2:S%=160
 3480 FOR Z%=0 TO 2:M%=?(&71+Y%+Z%)
 3490 A%=4^Z%:B%=2^(Z%*3)-(Z%=0)
 3500 IF (M% AND 2^(7-X%)) S%=S%+A%
 3510 IF (M% AND 2^(6-X%)) S%=S%+B%
 3520 NEXT:VDU S%:NEXT,
 3530 V%=V%+4-2*(L%=1):NEXT
 3540 ENDPROC
 3550 :
 3560 MODE7:REPORT:PRINT" at line ";ERL
 3570 *FX15,1
 3580 END
```

←42

Writing Your Own Compiler (cont'd)

```
 3970 IFT%=plus PROCLX:PROCfactor ELSEIF
T%=minus PROCLX:PROCfactor:PROCNEG ELSE
PROCfactor
 3980 ENDPROC
 3990 DEFPROCsound
 4000 PROCLX:PROCexp:PROCC(comma)
 4010 PROCLX:PROCexp:PROCC(comma)
 4020 PROCLX:PROCexp:PROCC(comma)
 4030 PROCLX:PROCexp:PROCSND:ENDPROC
 4040 :
 4050 DEFPROCmove:PROCLX:PROCVDL(25):PRO
CVDL(4):PROCexp:PROCVDD:PROCC(comma):PRO
CLX:PROCexp:PROCVDD:ENDPROC
 4060 DEFPROCdraw:PROCLX:PROCVDL(25):PRO
CVDL(5):PROCexp:PROCVDD:PROCC(comma):PRO
CLX:PROCexp:PROCVDD:ENDPROC
 4070 :
 4080 DEFPROCrep:LOCALL1%:PROCLX:L1%=P%
 4090 IFT%=eop ORT%=unt GOTO4110
 4100 PROCstate:IFT%=col ORT%=eoln PROCL
X:GOTO4090
 4110 PROCC(unt):PROCLX:PROCexp
 4120 PROCJMC(L1%):ENDPROC
 4130 :
 4140 DEFPROCwhi:LOCAL L1%,L2%:L1%=P%
 4150 PROCLX:PROCexp:L2%=P%:PROCJMC(0)
 4160 IFT%=eop ORT%=wnd GOTO4180
 4170 PROCstate:IFT%=col ORT%=eoln PROCL
X:GOTO4160
 4180 PROCC(wnd):PROCLX:PROCJMP(L1%)
 4190 V%=P%:P%=L2%:PROCJMC(V%):P%=V%
 4200 ENDPROC
```

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Editor: Phyllida Vanstone
Assistant Production Editor: Yolanda Turuelo
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.

Editorial Address
BEEBUG
PO BOX 50,
Holywell Hill,
St. Albans AL1 3YS

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address above. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.40 6 months (5 issues) UK ONLY
£11.90 UK - 1 year (10 issues)
£18 Europe, £21 Middle East
£23 Americas & Africa, £25 Elsewhere

BACK ISSUES
(Members only)

Vol	Single issues	Volume sets (10 issues)
1	90p	£8
2	£1	£9
3	£1.20	£11
4	£1.20	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues & Software Address

BEEBUG
PO BOX 109
St. Johns Road
High Wycombe HP10 8NP

Hotline for queries and software orders

St. Albans (0727) 40303
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

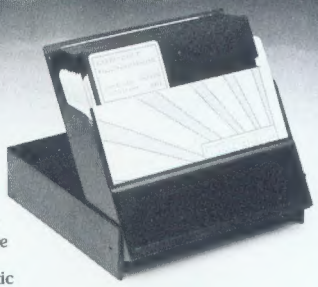
If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

DYNAMIC DISCS

TESTED BY BEEBUG

BEEBUG, the largest independent computer user group in the UK, offer 100% tested discs supplied by one of Britain's leading disc manufacturers.

10 FREE LIBRARY CASE

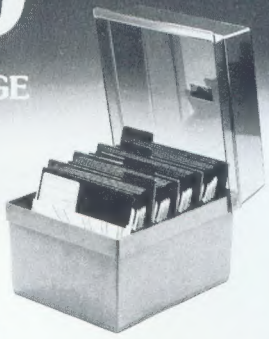


Orders for 10 discs are sent in black plastic library cases.

25 FREE STORAGE BOX

Orders for 25 are delivered in strong plastic Storage box with 4 dividers.

50 FREE STORAGE BOX



Orders for 50 are delivered in strong plastic Storage box with 4 dividers.

COMPARE PRICES

4 Types of Disc To Meet Your Exact Requirement

			48 TPI		DOUBLE DENSITY		
10	S/S	D/D	£14.90	10	D/S	D/D	£20.50
25	S/S	D/D	£34.90	25	D/S	D/D	£46.20
50	S/S	D/D	£59.30	50	D/S	D/D	£82.40
			96 TPI		DOUBLE DENSITY		
10	S/S	D/D	£20.50	10	D/S	D/D	£21.90
25	S/S	D/D	£46.20	25	D/S	D/D	£49.90
50	S/S	D/D	£82.40	50	D/S	D/D	£93.50

All prices include Storage Box, VAT and delivery to your door (UK)

Suitable for BBC Micro and all other computers using 5¼ inch discs including Atari and Commodore.

Fully Guaranteed — Not only by Beebug but by one of the UK's top disc manufacturers.

ORDER FORM

Official orders welcome
Access/Barclaycard
24 hrs line 0494 816666
Further information on helpline 0727 40303

Please supply me

@ £

@ £

@ £

Total cheque enc. £

Name

Address

To Beebugsoft, PO Box 109, High Wycombe, Bucks. HP10 8NP

BEEBUG SOFT

Magazine Cassette/Disc

JAN/FEB 1986 CASSETTE DISC CONTENTS

BEEBUG FILER – all three parts of the BEEBUG database manager combined

DISC RECOVERY – retrieve all those lost or corrupt disc files

THE EARTH FROM SPACE – a stunning view of the earth from any selected point in space

VIEWSHEET AUTOBOOT – a useful starter routine for spreadsheet users

FUZZY COMMANDS – make the Beeb recognise all your mis-typed commands

BASIC COMPILER – the second part of this program

BEEBUG WORKSHOP – proceedures and extended demo for flexible graphs

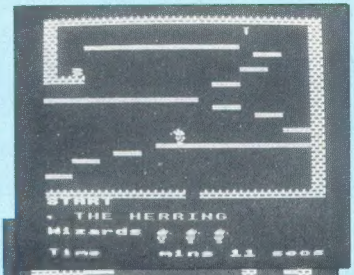
FIRST COURSE – two instructive examples of the use of BBC Basic's EVAL function

MANIC MECHANIC – a first rate and highly colourful "platform" style game

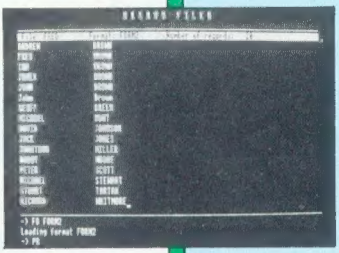
EXTRA FEATURES THIS MONTH

WIZZARD – another truly superb game combining the best elements of both arcade and adventure games in an enthralling and colourful extravaganza

MAGSCAN – data for this issue of BEEBUG (VOL. 4 No. 8)



Wizzard



BEEBUG Filer Part 3

Manic Mechanic

The Earth from Space

All this for £3.00 (cass) £4.75 (disc) + 50p p&p
Back issues (disc since Vol. 3 No. 1, cass since Vol. 1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

All subscription and individual orders to
BEEBUG, PO BOX 109, St. Johns Road, High Wycombe HP10 8NP

ISSN 0263-7561 Printed in England 1/2630